SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Kristian Faltis

Zagreb, 2012. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Bojan Jerbić, dipl. ing.
Tomislav Stipančić, dipl. ing.

Student:

Kristian Faltis

Zagreb, 2012. godina.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se profesoru Bojanu Jerbiću i mentoru Tomislavu Stipančiću na stručnoj pomoći u izradi ovog diplomskog rada.

Kristian Faltis

SADRŽAJ

POPIS SLIKA

SAŽETAK

Tekst sažetka (style: TEKST)

Ključne riječi: Open Inventor, vitualna stvarnost, računalne simulacije

Tema ovog diplomskog rada je Open Inventor knjižniza algoritama. Ona se koristi za kreiranje interaktivnih 3D aplikacija i virtualne stvarnosti.

U ovom diplomskom radu biti će obrađene sve tehničke i programske značajke Open Inventor knjižnice algoritama. Bit će opisana povijest Open Inventora, bit će rečeno što je Open Inventor, na koji način se koristi, od čega se sastoji i , na kraju , bit će prikazane dvije virtualne scene. Obje scene koriste Open Inventor knjižnicu algoritama. Scene sadrže robotske manipulatore i pripadajuću okolinu specifičnu za poslove robotskog sklapanja. Prva scena izrađena je u programu Visual Studio 2010 i napisana u C++ programskom jeziku. Druga scena je razvijena pomoću IvTune editora.

1. UVOD

1.1. Računalne simulacije

Računalne simulacije su pokušaj modeliranja stvarnog života ili hipotetske situacije na računalu, u svrhu proučavanja kako određeni sustavi funkcioniraju. Promjenom varijabli unutar simulacije može se pretpostaviti kako bi se određeni sustav ponašao u datoj situaciji. To je alat za virtualno istraživanje i proučavanje određenog sustava.

Računalne simulacije su postale korisne kod modeliranja mnogih prirodnih sustava u fizici, kemiji i biologiji, ljudskih sustava poput ekonomije i sociologije, kao i inženjerskih sustava kako bi dobili uvid u način rada određenog sustava.

Računalne simulacije se razvijaju paralelno s brzim razvojem računala, a prvu ozbiljniju primjenu doživljavaju za vrijeme drugog svijetskog rata u projektu „Manhattan“ kod modeliranja procesa detonacije nuklearne bombe. Bila je to simulacija koja je uključivala 12 sfernih objekata i koristila tada poznati „Monte Carlo“ algoritam. Od toga vremena primjena računalnih simulacija u sve je većem porastu.

Na tržištu vlada velika konkurencija, te je za opstanak nužan interdisciplinarni pristup temeljen na znanju i timskom radu, uz primjenu modernih metoda matematičkog modeliranja i računalnih simulacija. Prijašnji razvoj se uglavnom oslanjao na dugotrajna i skupa modelska i prototipna mjerenja, a siromašnije sredine su nužno kaskale za razvojem bogatijih. Danas se u velikoj mjeri skupa mjerenja zamjenjuju relativno jeftinim računalnim simulacijama (stvaraju se virtualni laboratoriji) temeljenih na matematičkim modelima što je podloga za kreativan pristup rješavanju problema, što siromašnijim sredinama omogućava ravnopravniju utrku na svjetskom tržištu.

Za primjenu modernih metoda nužno je educirati moderne inženjere koji znaju univerzalne principe, vladaju modernim računalnim metodama s kojima znaju rješavati složene tehničke probleme, koji će biti kreativni, skloni timskom radu, koji govore strane jezike itd.

Open Inventor knjižnica algoritama jedan je od primjera rada s računalnom grafikom. U ovom radu bit će opisane njezine tehničke i programske karakteristike. Isto tako, bit će opisan programski zadatak koji uključuje simulacijsko okruženje s šestoosnim robotskim manipulatorom zajedno s pripadajućom okolinom karateističnom za poslove robotskog sklapanja.

1.2. Povijest Open Inventora

Oko 1988. godine Wei Yan i Rick Carrey vodili su tvrtku SGI i pokrenuli su projekt nazvan IRIS Inventor. Cilj je bio stvaranje alata koji bi olakšao razvijanje 3D grafičkih aplikacija. Strategija se zasnivala na pretpostavci da se ne razvija dovoljno 3D aplikacija IRIS GL-om, a razlog tome bila je dugotrajnost izrade sučeljem na niskoj razini koje je osiguravao IRIS GL. Kada bi 3D programiranje bilo olakšano, kroz korištenje objektno orijentiranog API-a, veći broj ljudi bi stvarao 3D aplikacije i SGI bi imao koristi. Stoga, cilj je bio „jednostavnost korištenja“ i slogan „3D programiranje za sve ljude“ ubrzo je bio široko korišten.

Nešto kasnije je odlučeno IRIS GL zamijeniti sa OPEN GL-om. OpenGL (OGL) je knjižnica na niskoj razini koja uzima popise jednostavnih poligona i stvara sliku (renderira) što je brže moguće. Da bi bilo napravljeno nešto praktično, kao što je crtanje kuće, objekt se mora rastaviti u seriju jednostavnih OGL uputa koje se zatim šalju u proces renderiranja. Problem je što je OGL izvođenje iznimno osjetljivo na način na koji su ove upute poslane u sustav. Zahtijeva se od korisnika da zna koje upute poslati, kojim redom i prisiljava ih na pažljivu raspodjelu podataka da bi slanje objekata koji nisu vidljivi u završnoj slici bilo izbjegnuto. Za jednostavne programe ogromna količina programiranja mora biti napravljena samo da bi se započelo.

Open Inventor (OI) je napisan da bi ovaj problem bio riješen. On je koristio knjižnicu OpenGL. Objekti bi bili svrstavani po broju unaprijed pripremljenih oblika, kao što su kocke i mnogokuti, i zatim lako promijenjeni u nove oblike. Također su uključeni kontrolni objekti i drugi sustavi za stvaranje scene, što je olakšalo izvršavanje zadataka. Konačno, OI bio je u mogućnosti stvoriti svoj format datoteke za spremanje „virtualnih svjetova“ i koda. To je omogućilo automatsko spremanje i učitavanje projekata iz ovih datoteka. Osnovne 3D aplikacije bile su napisane tek u nekoliko stotina linija s OI-om i mogle su biti vezane tzv. „glue“ kodom.

Loša strana OI-a je što je sporiji od ručno napisanog koda, jer su 3D zadaci puno teži za dobru izvedbu bez miješanja podataka. Drugi realan problem je što OI može biti korišten samo u svom formatu, što prisiljava programere da pišu pretvarače za ulazak i izlazak iz Open Inventor sustava. Ti problemi će biti riješeni kroz godine razvoja API knjižnice.

Nekoliko godina nakon što je Open Inventor prebačen s „IRIS“ na „Open“ licencu za rad na razvoju kupile su još dvije tvrtke. Bio je licenciran za programere tvrtke Template Graphics Software (TGS) i Portable Graphics. TGS je kasnije kupio Portable Graphic i počeo prodavati licence koje su do tada bile besplatne. Tvrtka SGI je stala s razvojem Inventor projekta i uključila se u neke druge projekte s tvrtkom Microsoft.

Nakon mnogo godina potpune dominacije pod posjedničkim licenciranjem TGS-a, Open Inventor je bio ponovo pušten pod LGPL „open source“ licencom tvrtke SGI u kolovozu 2000. godine. U otprilike isto vrijeme, tvrtka SIM (Systems in Motion) napravila je vrlo sličnu API knjižnicu nazvanu Coin3D. Kongsberg grupa kasnije je kupila SIM i preimenovala ga u Kongsberg SIM.

Mercury Computer Systems je 2004. kupio tvrtku TGS koja je bila u njenom vlasništvu sve do lipnja 2009. godine kada je TGS ponovo postala samostalna tvrtka, nazvana Visualization Sciences Group (VSG) te se nastavila razvijati i podržavati Open Inventor.

„Open source“ verzija iz tvrtke SGI nije trenutno održavana i SGI nije pokazao nikakvu privrženost daljnjem razvoju knjižnice. Međutim, njezin „open source“ kod koristi se u nekoliko aktivnih projekata: projektu „open-source eXtensible Imaging Platform“ (tvrtke Siemens Corporate Research) i projektu „MeVisLab“ (tvrtke Traunhofer MeVis).

Kongsbergova SIM-ova Coin knjižnica i TGS-ov Inventor (danas VSG) još uvijek napreduju, aktivno se razvijaju i imaju dodatna brojna poboljšanja na originalni Inventor API , kao npr. opsežnu potporu za VRML standard.

Bez obzira na starost, Open Inventor API je još uvijek široko korišten za širok raspon znanstvenih i tehničkih vizualizacijskih sustava. Dokazao se dobro konstruiranim za djelotvoran razvoj jednostavnih, ali i kompleksnih softvera za 3D aplikacije. [1]

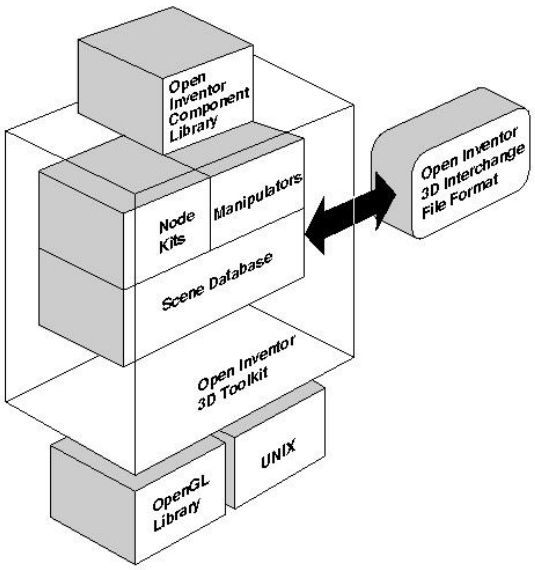
1.3. Open Inventor

Open Inventor je knjižnica objekata i metoda koje se koriste za izradu interaktivnih 3D grafičkih aplikacija. Koristi programske jezike C++, .NET, Java i C#. Inventor pretstavlja skup blokova koji omogućuje pisanje programa uz maksimalno iskorištavaje grafičke hardverske značajke i minimalan napor programiranja.

Program nudi knjižnicu objekata koja se može koristiti, mijenjati i proširivati kako bi se zadovoljile sadašnje i buduće potrebe. Open Inventor uključuje objekte zadužene za stvaranje slika, jednostavnih oblika, grupa, osvjetljenja, pogleda na scenu, boja, upravljačkih objekata poput interaktivnih manipulatora, upravljača smjera svjetla, ispitivača, preglednika i brojne druge objekte koji zajedno stvaraju scenu.

Program nudi ekonomičnost i učinkovitost cjelovitog objektno-orijentiranog sustava. Osim pojednostavljenog razvoja aplikacija, Inventor olakšava izmjenu podataka između aplikacija s vlastitim 3D formatom. Krajnji korisnici 3D programa mogu „izrezati“ i „zalijepiti“ 3D objekte i dijeliti ih između različitih programa na radnoj površini.

Kao što je prikazano na [Slika 1], "Open Inventor arhitektura", osnova Open Inventora je OpenGL i UNIX. Inventor sadrži objektno-orijentirana pravila temeljna na OpenGL knjižnici algoritama, pružajući model programiranja i korisničko sučelje za OpenGL programe.



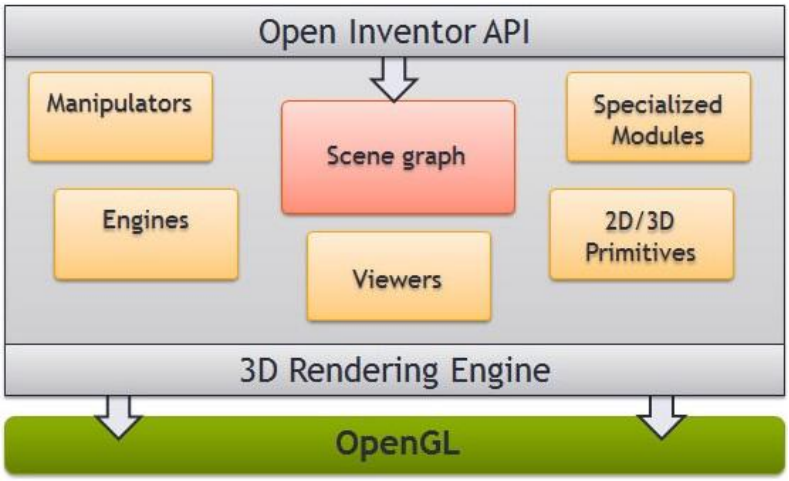
Slika 1 Open Inventor arhitektura

Open Inventor neovisan je o Windows sustavu, odnosno može biti korišten u bilo kojem operacijskom sustavu. Osim Windows operativnog sustava, Open Inventor pruža podršku za Linux, UNIX i Mac OS.

2. OPEN INVENTOR

2.1. Prednosti Open Inventor-a

Open Inventor ima brojne prednosti koje omogućavaju jednostavan način rada. Jednostavna shema rada s Open Inventor-om prikazana je na [Slika 2].



Slika 2 Open inventor način rada

Neke od prednosti Open inventora su [2]:

- 3D orijentirani objekti SDK (Software development kit)

Open Inventor je objektno-orijentirano sučelje s brojnim ugradbenim komponentama koje omogućavaju brzu primjenu i razvoj aplikacija.

Program izbjegava zahtjevno programiranje, složenu provedbu renderiranja i upravljanja algoritmima pomoću niže razine programiranja. Dakle, razvojni timovi će brzo izgraditi funkcionalne prototipe i implementirati profesionalne 3D aplikacija, ulažući više sredstava i vremena u ostale domene stručnosti i inovativnosti.

- Objektno-orijentirani 3D API / Graf scene

Open Inventor nudi sveobuhvatan objektno-orijentirani skup s više od 1.300 klasa koje su spremne za korištenje i integrirane u okvire prilagođene korisniku za brzi razvoj. Jednostavni graf scene omogućava grafičke programske obrazace koji su spremni za korištenje, a objektno-orijentirani dizajn potiče proširenja i prilagodbu kako bi zadovoljio specifične zahtjeve korisnika.

- Izbjegavanje niže razine API-a

Korištenje Open Inventora je u osnovi produktivnije zbog korištenja objektne-orijentiranosti. Pomoću dobro dizajniranih obrazaca, softver automatski koristi sve raspoložive značajke, automatski optimizira donošenje i provedbu uobičajeno viših razina funkcije i tako olakšava rad korisniku.

- Maksimalna fleksibilnost

Open Inventor je u potpunosti proširiva platforma i za razliku od drugih alata koji sakrivaju apstraktna sučelja hardvera on omogućava direktno povezivanje s nižom razinom API-a, ako je to potrebno. Dakle, bilo koji kod može biti integriran u OI arhitekturu. Može se razviti kod, kombinirajući svojstva više i niže razine programiranja unutar Open Inventor graf scene i na taj način implementirati novi okvir uz zadržavanje svih prednosti već postojećeg ili vlastitog razvoja.

- Napredni alati za razvoj

IvTune je grafički alat za izradu i podešavanje 3D aplikacija. Ugradiv je u bilo koju aplikaciju Open Inventora. IvTune predstavlja sinkronizirani pogled na scenu: 3D pogled, prozor „Tree View“ za uređenje, „Node Overview“ za postavljanje parametara scene itd. To omogućuje brzo podešavanje i ispravljanje pogrešaka u vrijeme izvođenja, a idealno nadopunjuje brojne primjere koda omogućene od strane Open Inventora.

- Platformno- nezavisan kod (Platform-independent code)

Cross-platforma u okviru Open Inventor-a omogućava programerima da dizajniraju podesive i interaktivne 3D aplikacije u Mac OS, UNIX, Linux i Microsoft Windows sustavima. Nastale aplikacije mogu biti 100% kompatibilne u izvornom kodu i potrebno ih je samo ponovo izgraditi (eng. „recompile“) da bi se mogle pokrenuti na drugim platformama.

2.2. Open Inventor formati datoteka

Open Inventor može čitati sljedeće formate [3]:

Geometrijski formati datoteka:

- .IV - Binary or ASCII
- .WRL - VRML 1.0, VRML97
- DXF
- OpenFlight
- Catia 5 (zahtjeva dodatnu licencu za korištenje)
- IGES (zahtjeva dodatnu licencu za korištenje)
- STEP (zahtjeva dodatnu licencu za korištenje)

Slikovni formati datoteka:

- GIF
- JPEG
- JPEG 2000
- PGX
- PNG
- PNM
- RGB
- Sun raster
- TIFF
- BMP (samo u Windows okruženju)

Formati koje Open Inventor stvara su samo geometrijski formati datoteka.

Open Inventor može stvoriti sljedeće formate [3]:

- .IV – Binary ili ASCII – Inventor datoteke
- .WRL - VRML 1.0

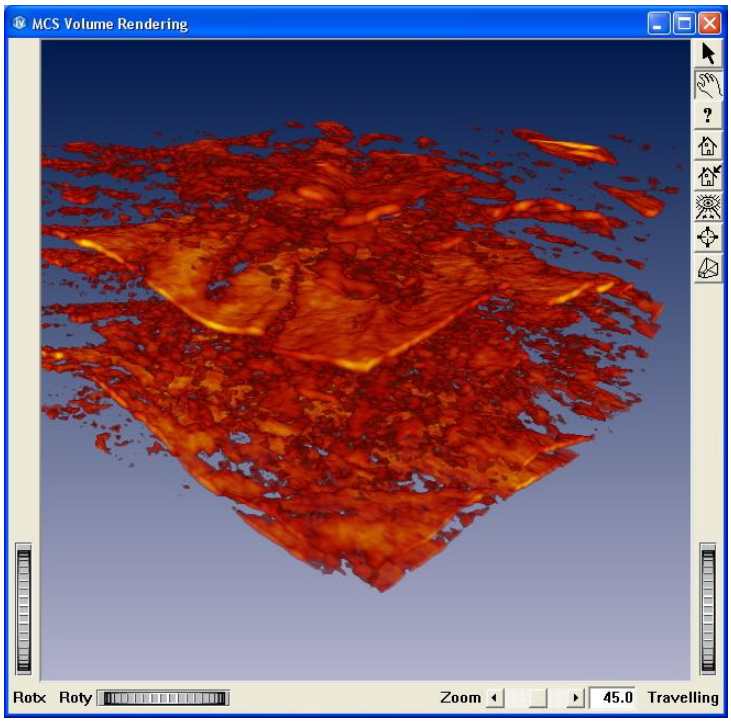
2.3. Proširenja (eng. „Extensions“)

Open invetor knjižnica algoritama ima ugrađena mnoga proširenja. Ona zapravo proširuju skup objekata, vrste objekata i tehnike programiranja koje su ranije napravljene.

Proširenja OIV-a su [4]:

- VolumeViz

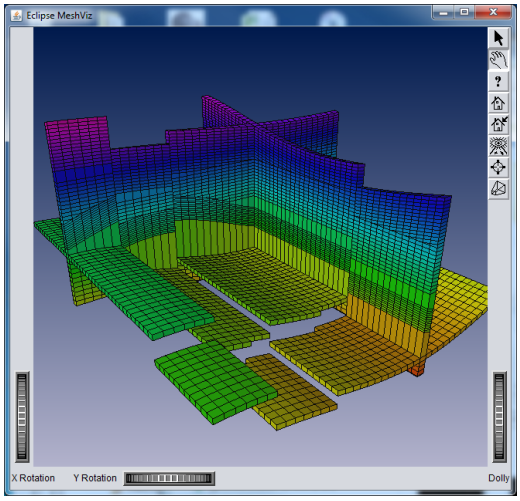
VolumeViz proširenje je alat za upravljanje i vizualizaciju podataka velikog opsega, bilo da su to veliki 3D objekti, brojne mikroskopske snimke, CT snimka ili podaci seizmičkih istraživanja. VolumeViz LDM (Local Data Manager) omogućava upravljanje podacima koje dozvoljava aplikacijama interaktivno stvaranje slike s podacima mnogo većih nego što je dostupna memorija sustava. Primjer scene gdje su korištene klase proširenja VolumeViz je [Slika 3].



Slika 3 VolumeViz primjer

- MeshViz

MeshViz dodaje podršku za upravljanje, stvaranje i vizualizaciju podataka u obliku mreže, bilo da je to analiza konačnih elemenata, ispitivanje dinamika fluida ili , primjerice, statike naftnih rezervoara. MeshViz XLM pruža jedinstveno API sučelje koji dozvoljava aplikacijama vizualizaciju bilo kojeg tipa mreže s bilo kojim tipom tipom ćelija (trokutići, kvadratići, mnogokuti...). Primjer proširenja MeshViz nalazi se na [Slika 4].



Slika 4 MeshViz primjer

- ScaleViz

ScaleViz dozvoljava aplikacijama povećanje broja prikaza, kapaciteta podataka i/ili stvaranje scene dodavanjem dodatnog GPU-a (Graphical Proces Unit). Zatim pruža podršku za realnije kreiranje virtualne stvarnosti i praćenje svih ulaza. Primjer proširenja ScaleViz nalazi se na [Slika 5].

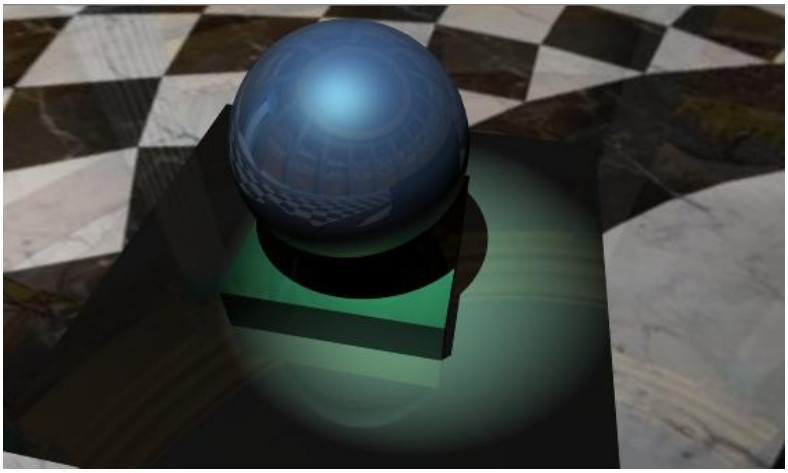
Slika 5 Primjer ScaleViz podrške za prikaz na više zaslona

- ReservoirViz

ReservoirViz LDM (Local Data Manager) je kombinacija mrežne vizualizacije s LDM upravljanjem podataka pomoću kojeg možemo kreirati, primjerice, interaktivnu navigaciju u sceni s jako velikim mrežnim prikazom naftnog rezervoara.

- DirectViz

DirectViz proširenje omogućava stvaranje 3D scene foto-realistične kvalitete koristeći OpenRTRT. Koristi se 3D opis scene kod koje su potrebne klase koje precizno određuju odnose svjetla i sjene, kontraste i stvaraju vjerne virtualne slike. Ovdje se koristi i tehnika „ray-tracing“ kod koje se stvara slika praćenjem svjetla kroz piksele. Primjer scene stvoren pomoću DirectViz proširenja nalazi se na [Slika 6].



Slika 6 Primjer DirectViz scene

- HardCopy

HardCopy proširenje pruža dva rješenja. Prvo je mogućnost pretvorbe pogleda na 3D scene u grupu 2D slika. Drugo je pretvorba 3D scene u oblik koji može biti izravno ugrađen u Adobe PDF datoteku i gledan korištenjem Acrobat Readera u 3D pogledu.

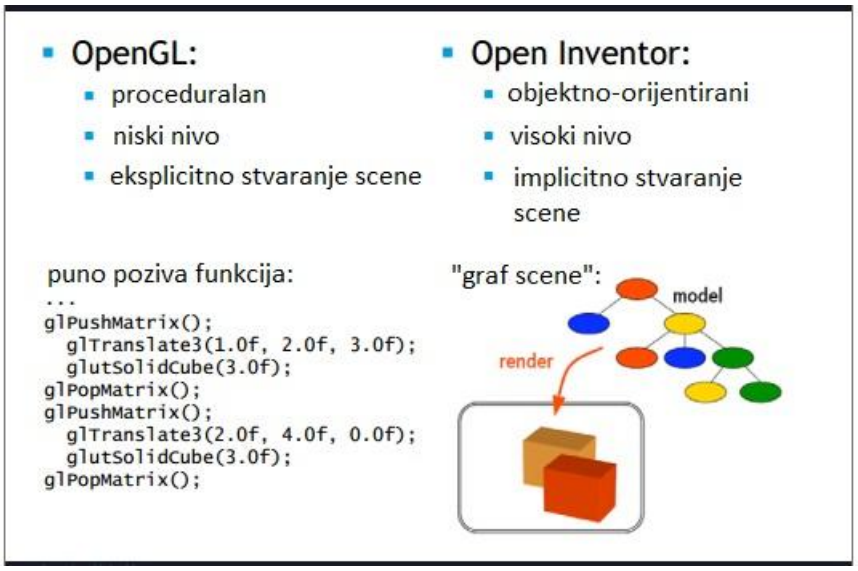
2.4. OpenGL

OpenGL ili Javna Grafička Knjižnica (eng. „Open Graphics Library“) je grafički 3D standard stvoren od strane Silicon Graphics tvrtke. Silicon Graphics je debitirala na svjetskom tržištu s moćnim Silicon Graphics računalima za obradu video zapisa tijekom osamdesetih i devedesetih godina prošlog stoljeća.

Standard se sastoji od preko 250 različitih funkcija koje defiraju načine crtanja kompleksnih 3D scena s jednostavnim slikama ili osnovnim geometrijskim elementima - kocka, trokut, piramida, itd. OpenGL je popularna biblioteka u industriji video igara, gdje se trenutno natječe s Microsoftovim Direct3D-om.

Open Inventor koristi OpenGL, standardno industrijsko sučelje, za stvaranje scene. OpenGL je ponekad korišten izravno za razvoj aplikacije. Stvaranje jednostavne scene u OpenGL-u u osnovi uključuje programiranje u programskom jeziku „C“ i puno poziva funkcije.

Zapravo, pravo stvaranje geometrije je „skriveno“ korištenjem funkcija iz GLUT knjižnice, ali u principu mora se „reći“ OpenGL-u točno kako treba kreirati svaku sliku scene. Usporedba OpenGL-a i Open Inventora prikazana je na [Slika 7].



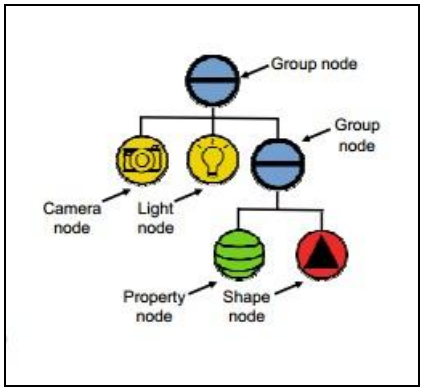
Slika 7 Usporedba OIV-a i OpenGL-a

Može se primijetiti da se prema pozivima funkcije OpenGL mora eksplicitno formulirati upravljanje, na primjer, spremanje i vraćanje trenutne transformacije da bi kocke mogle biti odvojeno pozicionirane.

Open Inventor je C++ (i Java i C#) knjižnica, pa se radi s „objektima“ koji upravljaju vlastitim svojstvima i dozvoljavaju jednostavnu promjenu tih svojstava. Gradi se „drvo“ (eng. „tree“) tih objekata nazvano „graf scene“ koji opisuje kako bi 3D scena trebala izgledati. Open Inventor se brine za stvaranje scene umjesto programera. Također ima prednost zbog poznavanja strukture scene s ciljem optimizacije. Međutim, interakcija Open Inventora i OpenGL-a je pažljivo definirana tako da je moguće definirati nove objekte izvedene iz postojećih objekata i također, ako je potrebno, miješati Open Inventor i OpenGL funkcije.[8]

2.5. Graf scene

Programiranja s Open Inventorom zapravo je stvaranje i modificiranje grafa scene. Kao što je rečeno, objekti u grafu scene formiraju strukturu drveta. Ti objekti zovu se čvorovi (eng. „nodes“). Oni pretstavljaju klase i funkcije iz Open Inventor knjižnice algoritama. Na [Slika 8] se može vidjeti primjer jednostavnog grafa scene i neke vrste čvorova koji će se koristiti za izradu grafa scene.



Slika 8 Graf scene

Još jedna vrsta objekta, zvana „akcija“ (eng. „action“), „hoda“ grafom scene, „posjećuje“ svaki čvor i pokreće određene akcije u čvoru. Primjeri akcija koji se koriste u Open Inventoru su:

- *Render* - stvaranje scene,
- *Pick* - odabir geometrije objekata u sceni i
- *Search* - pronalaženje specifičnog čvora ili čvorova.

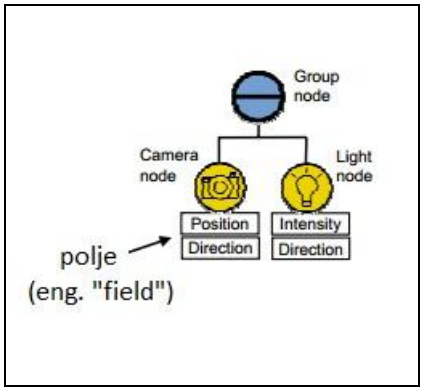
Kada se kreira graf scene on će biti odgovoran za pojavljivanje objekata i akcija koje ti objekti obavljaju u sceni. Kako bi korisnik izradio aplikaciju pomoću Open Inventora ključno je znati ispravno postaviti graf scene s njegovim čvorovima i akcijama. Open Inventor je odgovoran za stvaranje slike tih objekata, korištenjem OpenGL-a i grafičkog hardwarea on maksimizira kvalitetu izvođenja programa i jasnoću slike. [5]

2.6. Grupe čvorova

Kategorije (grupe) objekata grafa scene, ili čvorova, su [5]:

- *Group* - čvorovi koji definiraju strukturu (hijerarhiju) grafa scene
- *Data* - čvorovi koji čuvaju podatke o objektima (ili pokazuju na njih)
- *Transform* - čvorovi koji služe za skaliranje, rotiranje i pozicioniranje geometrije
- *Property* - čvorovi definiraju svojstva objekata kao što su boja, veličina fonta, itd.
- *Shape* - čvorovi koji definiraju stvarnu geometriju koriste se ostalim čvorovima kako bi definirali svojstva, poziciju i ostale podatke

Svi čvorovi spremaju svoje vrijednosti (iznose) u posebne varijable zvane „polja“ (eng. „fields“) . Na primjeru na [Slika 9], čvor kamere (Camera node) može imati svojstvo „Pozicija“ (Position) i „Smjer“ (Direction), dok čvor svjetla (Light Node) može imati svojstva „Intenzitet“ (Intensity) i „Smjer“ (Direction) .



Slika 9 Polja (eng. „Fields“)

Kako bi se kreirao čvor u C++ -u potrebno je koristiti sljedeću sintaksu:

```
SoCylinder *pNode = new SoCylinder();

pRoot->addChild(pNode);
```

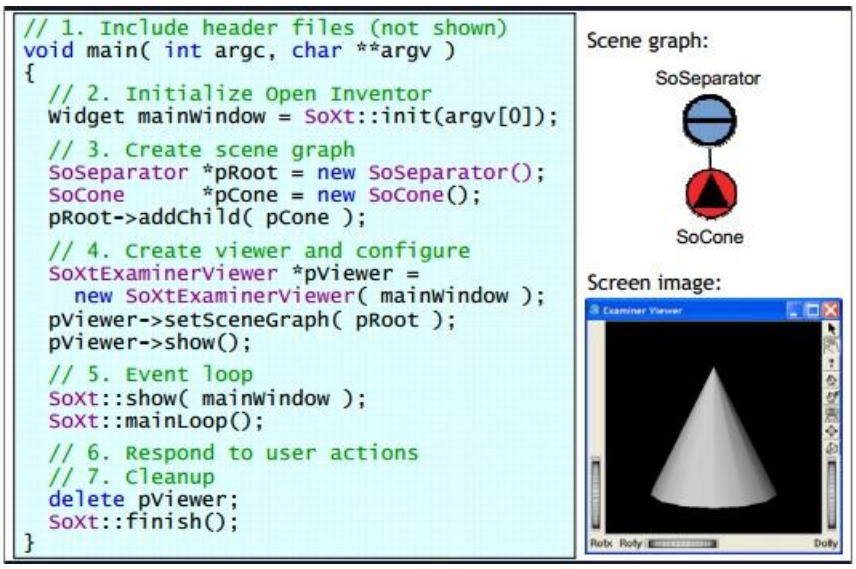
Ovo je generalna sintaksa koja kreira čvor napisana u C++ - u, u drugim programskim jezicima (.NET, Java, C#) imena klasa su iste, a sintaksa je vrlo slična. Klasa *SoCylinder* je klasa grupe *Shape* i ona kreira stožac u grafu scene.

3. PROGRAMIRANJE S OPEN INVENTOROM

3.1. Način rada

1. Pokrenuti program koji može koristiti Open Inventor knjižnicu algoritama (npr. Visual Studio 2010)
2. Uključiti C++ datoteke zaglavlja (eng. „header files“)
3. Izraditi Open Inventor graf scene
4. Izraditi preglednik kojim možemo pratiti našu scenu
5. Pokrenuti petlju i prikazati prozor scene
6. Prema potrebi dodavati, brisati i modificirati objekte unutar scene
7. Izlazak iz programa

Sve Open Inventor aplikacije koriste sličnu rutinu prikazanu prethodno. Na [Slika 10] prikazan je program napisan u C++ - u, struktura grafa scene u obliku drveta i prozor koji se generira i koji omogućava pogled na scenu.



Slika 10 Primjer programa u OIV-u

Na [Slika 10] desno-gore se nalazi graf scene koji stvara jednostavnu scenu sa stožcem u sredini preglednika. Prikaz prozora sa stožcem nalazi se na [Slika 10] dolje-desno. Lijevo se nalazi programski kod koji služi za izradu scene. Open Inventor programski kod za izradu jednostavnog stožca je kratak u odnosu na istu aplikaciju koja bi bila napisana u OpenGL knjižnici algoritama.

U nastavku će biti objašnjeni svi koraci prikazani na [Slika 10].

1. Prvi korak je uključivanje datoteka zaglavlja pomoću kojih se koriste klase koje pruža Open Inventor.

2. U drugom koraku koristi se `SoXt::init` metoda kako bi se inicijalizirala Open Inventor knjižnica algoritama i pokrenuo prozor koji korist naša aplikacija. Ovdje se koriste *SoXt* grupa klasa koja se obično koristi u X11/Xt/Motif sučelju. Open Inventor automatski mijenja tu grupu klasa u ekvivalentna *SoWin* imena kako bi aplikacije mogle biti korištene za izradu alikacije na Windows platformi. To nam omogućava da radimo na više platformi s istim programskim kodovima i klasama.

3. U trećem koraku kreira se graf scene. U ovom jednostavnog grafu scene nalazi se samo grupacijski čvor (Group Node), koji se naziva *SoSeparator*, i čvor za oblikovanje (Shape Node) koji se naziva *SoCone*. Hijerarhija između čvorova u grafu scene definira se pomoću `addChild()` metode. Ako čvor za grupiranje (*SoSeparator*) bude vezan za neku akciju i čvor za oblikovanje (*SoCone*) će biti vezan za neku akciju. U većini slučajeva graf scene sadržava i čvor za osvjetljenje i čvor kamere koji imaju polja sa parametrima osvjetljenja i parametrima kamere npr. polje koje sadrži varijable o količini osvjetljenja ili polja koja sadrže parametre o položaju i kutu kamere. U ovom slučaju oni nisu inicijalizirani pa graf scene sadrži unaprijed određene parametre osvjetljenja i parametre kamere koja prikazuje scenu.

4. U četvrtom koraku kreira se instanca za najčešći tip preglednika za prikaz scene tzv. „Examiner Viewer“. Postoje i drugačiji tipovi preglednika s drugačijim sučeljem. Druge vrste preglednika imaju i drugačiji „odgovor“ na korištenje kompjuterskog miša. „Examiner Viewer“ vrlo dobro prikazuje objekte i grupe podataka. On daje mogućnost korisniku da se pomiče ili rotira oko središta preglednika (ili bilo koje druge točke u pregledniku). Čvor kamere i čvor osvjetljenja su unaprijed određeni kao „dijete“ (eng. „child“) grupacijskog čvora. Stoga će se svaka akcija, koju pridodamo grupacijskom čvoru, odnositi i na čvor osvjetljenja i čvor kamere.

5. U koraku pet zatvara se programska petlja i program se izvršava sve dok ga korisnik ne prekine.

6. U koraku šest nije ništa napisano, ali bi se na tom mjestu programskog koda mogla dodati veza između korisnika aplikacije i same aplikacije. Korisnik bi mogao dodati određenu klasu koja bi utjecala na scenu npr. pritiskom određene tipke stožac u sceni se rotira ili translatira.

7. U sedmom koraku nalazi se dio koda koji prekida program na zahtjev korisnika.

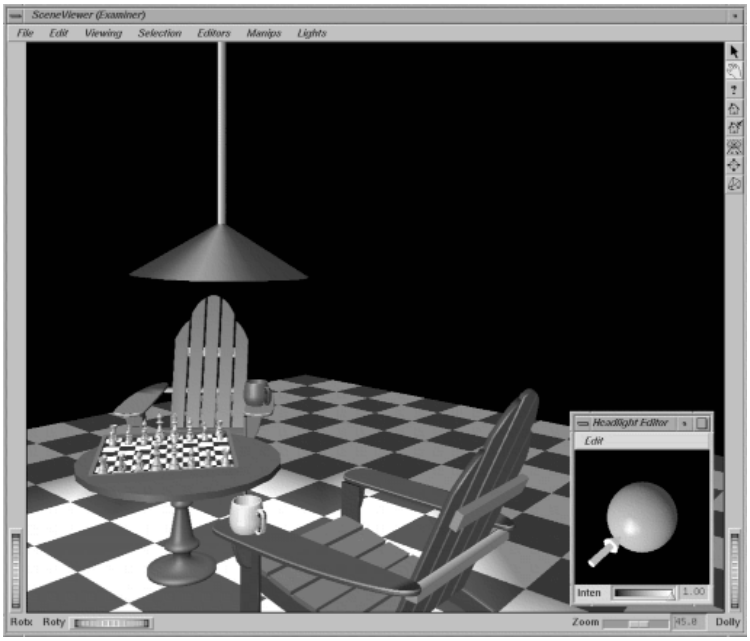
3.2. *Viewer* klase

Jedna od moćnih opcija Open Inventora su klase koje generiraru pogled na scenu. Ove klase služe za izradu 3D prozora u Window sustavu ili drugim operativnim sustavima, olakšavajući postavljanje jednog ili više 3D prozora u korisničkom sučelju. Postoje posebne verzije ove klase za X11/Motif, Windows primitive, Qt, Wx, itd.

Viewer klase imaju velike mogućnosti intergracije i pružaju poglede s mnogo fukcionalnosti. Programeri koji rade direktno s OpenGL knjižnicom algoritama moraju potrošiti više vremena na implementaciju ili traženje ovih klasa na Internetu . Open Inventor klase za generiranje pogleda na scenu su puno jednostavnije u odnosu na OpenGL kod kojih se moraju izvršiti sljedeći koraci:

- postavljanje i inicijalizacija OpenGL prozora
- programiranje korištenja kompjuterskog miša s ciljem navigacije u 3D sceni i
- podrška za stereo, postavljanje slike na cijelom zaslonu, antialiasing i dr.

Na [Slika 11] prikazan je prozor koji je generiran najčešće korištenom klasom za pogled na scenu, tzv. „Examinar viewer“. [5]



Slika 11 „Examinar viewer“ prozor

3.3. Polja

Već je bilo riječi o tome kako svaki čvor u Open Inventoru ima svoja polja. Polja nisu samo članovi varijabli. Ona su „pametni“ spremnici podataka s puno zanimljivih i korisnih sposobnosti. Neki od njih, kao STL spremnici, automatski upravljaju memorijom i tako čuvaju sve podatke koji se spreme u njih. Druge značajke su:

- polja mogu emitirati svoj sadržaj na zahtjev, dozvoljavajući tako grafu scene da bude lako modificiran i vrlo fleskibilan,
- polja mogu biti spojena s drugim poljima i širiti promjene kroz graf scene,
- polja mogu dati obavijest kada je sadržaj promijenjen.

Postoje dvije tipa polja: polja s jednom vrijednosti čija imena počinju sa *SoSF* i polja s više vrijednosti čija imena počinju sa *SoMF*. U te dvije grupe postoje odvojene klase za jednostavne tipove podataka kao sto su `int` i `float`, česte grafičke tipove kao što su točke i vektori, kao i kompleksne tipove kao što su slike. U nastavku će biti napisani i objašnjeni primjeri *SoSF* i *SoMF* polja.

Primjer korištenja *SoSF* polja je prikazan na [Slika 12]:

```
SoCylinder *pNode = new SoCylinder;
pNode->radius.setValue( 5.0 );
float r = pNode->radius.getValue();
```

Slika 12 Primjer SoSF polja

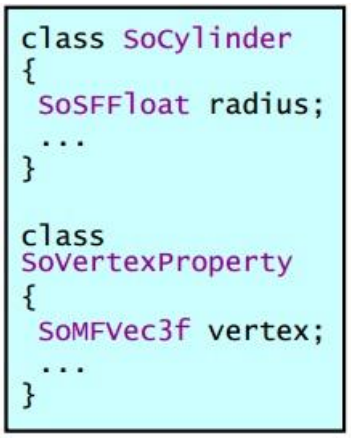
Primjer korištenja *SoMF* polja je prikazan na [Slika 13]:

```
SoVertexProperty *pNode = new SoVertexProperty;
pNode->vertex.set1Value( 0, SbVec3f(2,3,4) );
SbVec3f coord = pNode->vertex[0];
```

```
pNode->vertex.setValues( start, num, valuesArray );
pNode->vertex.setValuesPointer( num, valuesArray );
```

Slika 13 Primjer SoMF polja

Da bismo razumjeli napisane primjere prikazana su svojstva klasa *SoCylinder* i *SoVertexProperty* na [Slika 14]:



Slika 14 Svojstva klasa *SoCylinder* i *SoVertexProperty*

Iz [Slika 14] može se vidjeti da *SoCylinder* klasa, koja služi za oblikovanje stožastih objekata u sceni, ima `float` varijablu s jednom vrijednosti nazvanu „radius“ koje određuje radijus cilindra. Vrijednost radijusa je polje čvora *SoCylinder*.

U primjeru na [Slika 12] može se vidjeti `.setValue` i `.getValue` metoda. Pomoću prve metode postavlja se radijus baze stožca, dok druga metoda inicijalizira varijablu *r* koja je tipa SoSF `float` i sadrži vrijednost radijusa.

U primjerima na [Slika 13] i [Slika 14] se vidi da *SoVertexProperty* čvor ima polje s više vrijednosti koje sadrži „Vec3f“ objekte, u ovom slučaju koordinate vrhova koje definiraju neki oblik. Sintaksa je nešto drugačija kod polja s više vrijednosti, metoda `.setValue` i ovdje postoji i određuje prvu vrijednost u polju. `Set1Value` uzima index vrijednosti i koristi ga u naredbi `set`. Zagrada se nadglašava da bi se dobila vrijednost određenog indexa.

Mnogo vrijednosti može biti određeno s jednim `setValue` naredbom. Ovo je vrlo važno za zadatke kod kod kojih se postavlja velik broj vrijednosti. Dobro je znati da sve ove metode rade kopiju podataka aplikacije. Također se može koristiti `setValuesPointer` da se kaže polju da koristi vrijednosti koje su sačuvane u memoriji aplikacije, bez potrebe za kopiranjem. [5]

3.4. Čvorovi grupe *Property*

Mogu se koristiti različite vrste čvorova grupe *Property* da se odredio izgled i ponašanje čvorova u grafu scene.

Najčešće korišteni čvorovi grupe *Property* su:

- Čvor *SoMaterial*
    - *SoMFColor diffuseColor* = R,G,B float u rasponu od 0..1
    - *SoMFFloat transparency* = 0 neproziran .. 1 proziran

Čvor *SoMaterial* sadrži nekoliko polja koja su izvedena iz OpenGL knjižnice algoritama. Najčešće korištena polja su *diffuseColor* i *transparency*. Polje *diffuseColor* sadrži `float` vrijednosti crvene, zelene i plave boje u rasponu od 0 do 1. Rezultantna boja na sceni nastaje miješanjem sve tri nijanse zadanih boja. Ovo nam je potrebno ukoliko želimo stvoriti scenu sa više od 8 bitova boje po određenoj komponenti. Prozirnost je određena s `float` vrijednosti od 0 do 1 i ona određuje stupanj prozirnosti objekta na kojeg ovaj čvor utječe. Ako je vrijednost 0 to znači potpunu neprozirnost, dok 1 znači potpunu prozirnost.
Može se primijetiti da su i *diffuseColor* i *transparency* polja s višestrukim vrijednostima (*SoMF* tip polja) pa se mogu primijeniti različite boje i stupnjevi prozirnosti za različite dijelove objekta na koji se odnose.

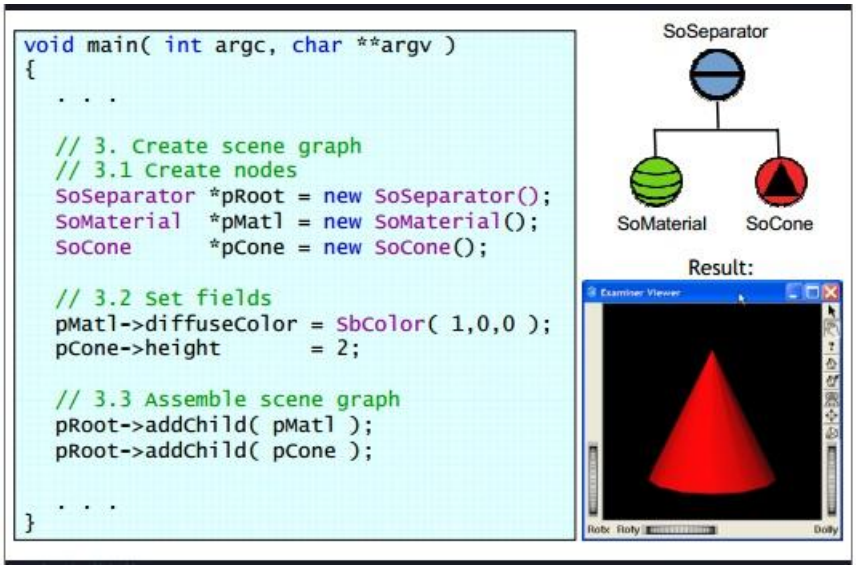
- Čvor *SoDraw*
    - *SoSFEnum style* = poligoni, linije, točke, …
    - *SoSFFloat lineWidth* = širina u pikselima

Čvor SoDraw služi za crtanje linija, točaka i različitih poligona, on sadrži polje *style* koje kontrolira način na koji čvor SoDraw crta objekte u sceni. Polje lineWidth koje je također sadržano u čvoru SoDraw utječe na debljinu linija objekata koje generira čvor SoDraw.

- Čvor *SoFon*t
    - *SoSFName name* = ime fonta i stil fonta
    - *SoSFFloat size* = veličina fonta

Još jedan primjer je SoFont čvor čija polja određuju font i veličinu korištenu za stvaranje teksta u sceni. [6]

Na [Slika 15] prikazan je program za izradu crvenog valjka koristeći čvor *SoMaterial*.



Slika 15 Primjer grafa scene sa čvorom *SoMaterial*

Na početku našeg programa inicijalizirali smo 3 čvora:

- *SoSeparator*
- *SoMaterial*
- *SoCone*

U nastavku polje *diffuseColor* čvora *SoMaterial* se postavlja u vrijednost `SbColor( 1,0,0 )`. To rezultira prikazom stožca u crvenoj boji. Također se postavlja polje *height* čvora *SoCube*, koje određuje visinu stožca. Ovo polje je postavljeno samo za ilustraciju jer Open Inventor automatski postavlja to polje u vrijednost 2 kad inicijaliziramo čvor *SoCube*. Može se primjetiti da se čvor *SoMaterial* stavlja prije čvora *SoCube* jer čvorovi grupe *Property* (*SoMaterial*) utječu na čvorove jedino „nizvodno“ tj. na sve čvorove koji su u programskom kodu inicijalizirani kasnije. Prikaz scene na ekranu i prikaz grafa scene nalaze se na [Slika 15] desno.

Nije važno kojim se redom postavljaju čvorovi. Raspored u prvom dijelu programa je samo jedna od mogućnosti. Međutim, kada se pozivaju funkcije `addChild` na kraju programa [Slika 15] čvor *SoMaterial* mora se nalaziti iznad čvora *SoCube* kako bi stožac mogao biti crvene boje tj. inicijalizacija čvora je učinjena pozivom funkcije `addChild`. [8]

Ostali korisni čvorovi grupe *Property* su:

- *SoDirectionalLight* – čvor za osvjetljenje scene
- *SoShadowGroup* – čvor za stvaranje sjene objekata u sceni

- čvorovi korišteni za teksturu:
  - *SoShadowGroup*
  - *SoCubeMapTexture*

- napredni čvorovi za kontrolu svjetla i sjene:
  - *SoVertexShader*
  - *SoGeometryShader*
  - *SoFragmentShader*

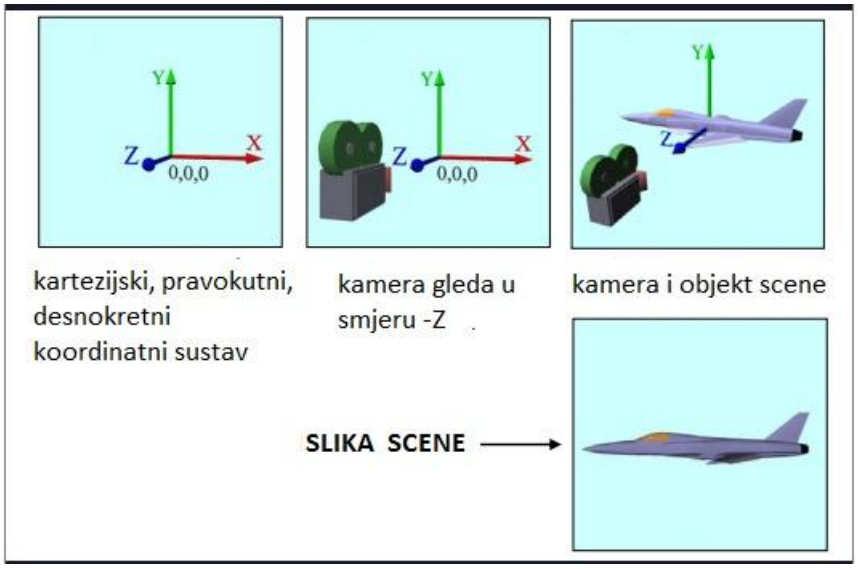
To su samo neki od brojnih čvorova koje Open Inventor pruža. Važno je još jednom napomenuti kako se svi čvorovi grupe *Property* moraju u programskom kodu nalaziti iznad čvorova grupe Shape (*SoCube, SoCone*...) kako bi se vrijednosti njihovih polja mogle pridjenuti objektima scene koje stvara grupa *Shape* (npr. kocka, stožac, kugla...) [6]

3.5. Pozicioniranje u sceni

3.5.1. Koordinatni sustav

Prilikom pozicioniranja potrebno je znati na koji način Open Inventor postavlja koordinatni sustav scene. Potrebno je razjasniti kako je 3D prostor definiran.

Open Inventor koristi isti koordinatni sustav kao OpenGL, koji je nazvan kartezijski, pravokutni, desnokretni koordinatni sustav, kao što je prikazano na [Slika 16].



Slika 16 Koordinatni sustav scene

Zadana svojstva čvora za postavljanje kamere definiraju sam pogled 3D prostora kao OpenGL. Koordinata +X je desno, +Y je „gore“ i +Z pokazuje prema korisniku [Slika 16].

To se zove „globalni“ koordinatni sustav, što je suprotno „lokalnom“ koordinatnom sustavu povezanom sa specifičnom geometrijom.

Kamera je namještena da vidi domet -1 do 1 u X,Y i Z, ali scena vjerojatno ima drugačiji opseg u 3D. Kamera se može u početku postaviti tako da je cijela scena vidljiva. U mnogim slučajevima je to automatski ili bar vrlo lako.

Open Inventor omogućava *viewAll* metodu koja automatski pomiče kameru tako da je cijela scena vidljiva. Zapravo, ako korisnik dozvoli automatsko kreiranje čvora kamere, automatski će se pozvati *viewAll* metoda pomoću `setSceneGraph` sintakse koju korisnik mora koristiti da bi inicijalizirao pogled na scenu. [8]

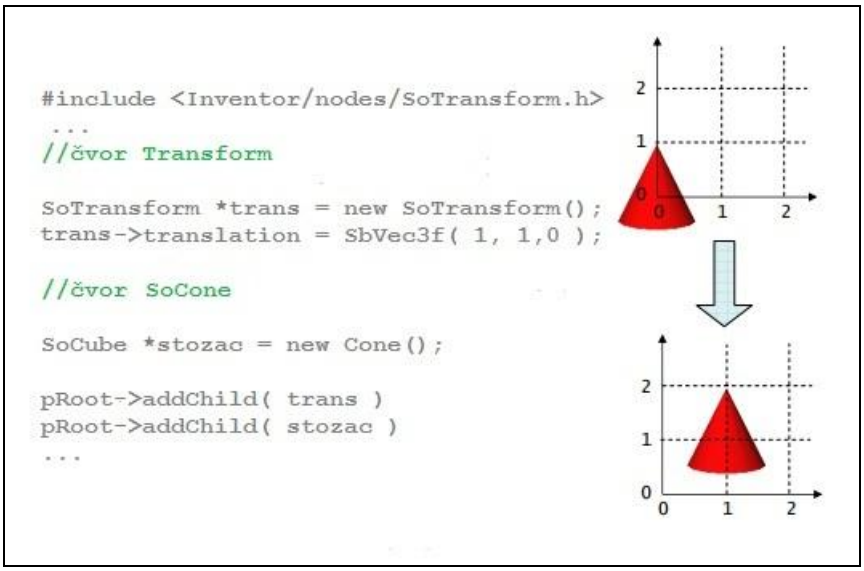
3.5.2. Čvorovi grupe *Transform*

Kako bi mogli pozicionirati objekte scene u željenu poziciju i orijentaciju koriste se čvorovi grupe *Transform*. Čvorovi grupe *Transform* služe kako bi se utjecalo na položaj i veličinu objekata u grafu scene.

Najčešće korišteni čvorovi ove grupe su:

- *SoTransform* - ima utjecaj na translaciju, rotaciju i veličinu objekata u sceni
- *SoTranslation* - translatira objekte u smjeru osi X,Y i Z
- *SoRotation* - rotira objekte oko osi X,Y i Z
- *SoScale* - utječe na veličinu objekata u sceni

Čvor *SoTransform* sadrži polja koja omogućuju odvojeno određivanje vrijednosti skaliranja, rotacije i translacije. Također se mogu koristiti pogodni čvorovi kao što je *SoTranslation* koji sadrže samo polje translacije. Ne postoji razlika u izvođenju, ali završni kod može biti malo više održiv zbog toga što je kod čvora *SoTranslation* samo translacija moguća za dotični objekt. Na [Slika 17] je prikazan primjer pozicioniranja stožca pomoću čvora SoTransform.



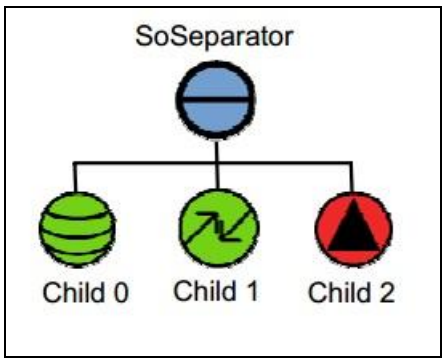
Slika 17 Primjer pozicioniranja stožca

Kako i kod grupe *Property*, čvorovi grupe *Transform* moraju se inicijalizirati prije čvorova grupe *Shape* tj. sintaksa `pRoot->addChild(trans)` ide iznad `pRoot->addChild(stozac)`. [8]

3.6. Čvorovi za grupiranje

Open Inventor sadrži čvorove za grupiranje ostalih čvorova. Svaki čvor za grupiranje je izveden iz grupe *SoGroup* i sadrži nula ili više čvorova. Čvorovi unutar čvora za grupiranje su tzv. „child“ čvorovi dotičnog čvora. Čvorovi moraju biti poredani točno određenim redoslijedom. Redoslijed čvorova utječe na odnose među njima, čvorovi koji utječu na druge čvorove moraju biti inicijalizirani prije čvorova na koje utječu.

Na [Slika 18] nalazi se graf scene koji prikazuje čvor za grupiranje *SoSeparator* i tri „child“ čvora unutar njega.



Slika 18 Primjer SoSeparator čvora

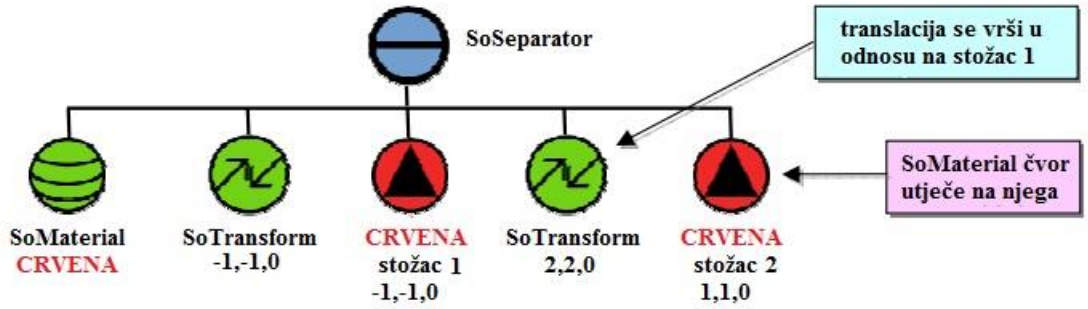
Najčešće korišten grupirajući čvor je *SoSeparator*. Ovaj čvor se koristi kako bi se grupiralo više drugih čvorova i na taj način ih odvojilo od ostatka grafa scene. Na taj način ne dozvoljava se čvorovima u drugom čvoru *SoSeparator* da utječe na njih. Čvor koji bi stajao izvan čvora *SoSeparator* i nalazio se hijererhijski iznad drugog čvora *SoSeparator* i dalje bi imao utjecaj na taj čvor i sve čvorove unutar njega.

Još jedan koristan grupirajući čvor je *SoSwitch*. Čvor *SoSwitch* može imati više „child“ čvorova kao i bilo koji drugi grupirajući čvor, ali se sintaksom `whichChild` određuje da li obuhvaća nulu, jedan ili sve „child“ čvorove koje se nalaze u njemu. [5]

3.7. Hijerarhija čvorova

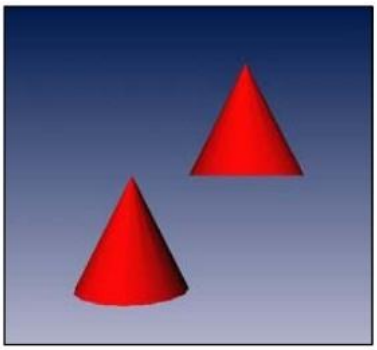
Prilikom programiranja sa Open Inventorom knjižnicom algoritama potrebno je poznavati hijerarhiju unutar grafa scene. Hijerarhija određuje na koji način pojedini čvorovi utječu jedni na druge.

Open Inventor kreira hijerarhiju na dva načina. Prvi način je grupiranje čvorova pomoću čvorova iz grupe *SoGroup*. Na taj način stvara se izolirani sustav čvorova koji sadrži svojstva, poziciju i geometriju objekata koji su prikazani u sceni. Drugi način je tzv. „vertikalna hijerarhija čvorova“ tj. čvorovi koji su inicijalizirani prije ostalih čvorova odnose se na sve čvorove inicijalizirane kasnije u Open Inventor sintaksi koda. Primjer čvorova koji su grupirani čvorom SoSeparator prikazan je na [Slika 19].



Slika 19 Čvorovi grupirani u čvoru SoSeparator

Prikazana je scena koja na ekranu prikazuje dva crvena stožca [Slika 20]. Kada se kreira sintaksa u C++ kodu prvo je potrebno inicijalizirati čvor *SoSeparator* i unutar njega 5 „child“ čvorova [Slika 19]. Prvo kreiramo čvor *SoMaterial* koji daje boju svim objektima ispod sebe (stožac 1 i stožac 2), zatim kreiramo čvor *SoTransform* koji translatira objekte inicijalizirane ispod sebe, a to je *stožac 1* i *stožac 2*. Drugi čvor *SoTransform* translatira objekte „ispod“ sebe, a to je *stožac 2*. Vrlo je važno kojim redom kreiramo čvorove kako bi se svojstva i vrijednosti pojedinih čvorova mogle odnositi na druge čvorove. [8]



Slika 20 Scena sa dva crvena stožca

3.8. Čvorovi za manipulaciju

Grupa *SoEngine* sadrži čvorove koji su odgovorni za promjenu u slici koju je moguće stvorili u Open Inventoru. Očit primjer korištenja grupe vidljiv je u animacijama, ali to je samo jedan od načina korištenja.

Čvorovi grupe *SoEngine* se mogu koristiti kao odgovor na vanjske naredbe, kao što su korištenje miša ili tipkovnice. Za ostvarivanje pune snage *SoEngine* klasa potrebno je dobro poznavati C + + programiranje, ali postoje neki jednostavni animacijski efekti koji se mogu stvoriti u IvTune editoru o kojem će biti riječi u kasnijem tekstu.

Osnovni pojmovi kod korištenju *SoEngine* čvorova su:

- Imaju jedan ili više ulaznih polja. Moguće je spojiti ulaz s vrijednostima na nekom drugom polju grafa scene. Kada se napravi promjena na tim poljima, ulaz čvorova unutar grupe *SoEngine* se mijenja. SoEngine čvorovi tada preduzimaju neke akcije kao odgovor na promjene ulaza. Svako polje ulaza može se priključiti samo na jednu stvar.

- Ima jedno ili više izlaznih polja. To je mjesto gdje su akcije *SoEngine* čvorova vidljive. Mogu se spojiti izlazna polja s drugim dijelovima grafa scene, tako da se trenutno stanje grafa scene mijenja onda kada se mijenja izlaz čvora *SoEngine*. Svako izlazno polje može biti spojeno na više ulaza.

- U praksi postoji mali broj čvorova *SoEngine* grupe koji služe za korištenje u animacijama, i njima se obično upravlja (izravno ili neizravno) pomoću unutarnjeg sata računala. Drugim riječima, za animacije se često povezuje ulazno polje čvora *SoEngine* grupe s nekim drugim poljem u graf sceni koje na neki način reagira na promjene u vremenu.

U daljnjem tekstu navedene su samo neke od vrsta čvorova SoEngine grupe, dostupnih u Open Inventoru, jer se samo oni mogu učinkovito koristiti bez naprednog poznavanja C++ programiranja.

Čvorovi *SoEngine* grupe su:

- *ElapsedTime* – ovo su vrlo jednostavne animacije motora, imaju polje za unos pod nazivom *timeIn* i izlazno polje *timeOut*. Također postoji polje brzine koje se može koristiti za kontrolu brzine izlaza. Uobičajena brzina ima vrijednosti između 0.0 i 2.0, sa 1.0 kao početnom zadanom vrijednošću, dok veći broj rezultira bržim izlazom. Negativne vrijednosti mogu se koristiti za promjenu smjera. Povezivanjem polja *timeIn* s njegovom promjenom, polje *timeOut će* se također promijeniti.

Budući da je svrha čvora *ElapsedTime* ukazati na količinu vremena koje je prošlo od početka rada, najjednostavniji i najočitiji način spajanja polja je spojiti globalno polje *RealTime* i *timeIn* čvora ElipsedTime. Ako ne definiramo drugačije, ova dva polja su automatski spojena.

- *TimeCounter* - čvor koji proizvodi ciklički izlaz. On počinje računanje od početne do krajnje vrijednosti, generiranjem izlaza za vrijeme računanja, nakon postizanja krajnje vrijednosti, čvor započinje svoj rad ispočetka.

- *Calculator* – radi točno ono što mu samo ime govori, on izračunava stvari. Ovi čvorovi sami po sebi ne služe za animaciju, ali su korisni za kontrolu ponašanja drugih čvorova, omogućavajući korisniku da koristi jednostavne matematičke izraze za transformaciju vrijednosti povezanih s poljem za unos podataka, oni će u željenom obliku biti poslani u izlazno polje *Calculator* čvora,a time i na sve čvorove spojene na ta izlazna polja.

- *Rotor Engines* – čvor koji okreće ostale objekte. Točnije, oni primjenjuju rotaciju na objekte kontinuirano tijekom vremena. Oni se ponašaju baš kao čvor *SoRotation*, osim činjenice da se kut rotacije mijenja kontinuirano tijekom vremena. Ovo je vrlo jednostavan čvor jer ne treba nikakvo povezivanje s poljima da bi mogao raditi. Korišten je u ovom diplomskom radu za animiranje šestoosnog manipulatora.

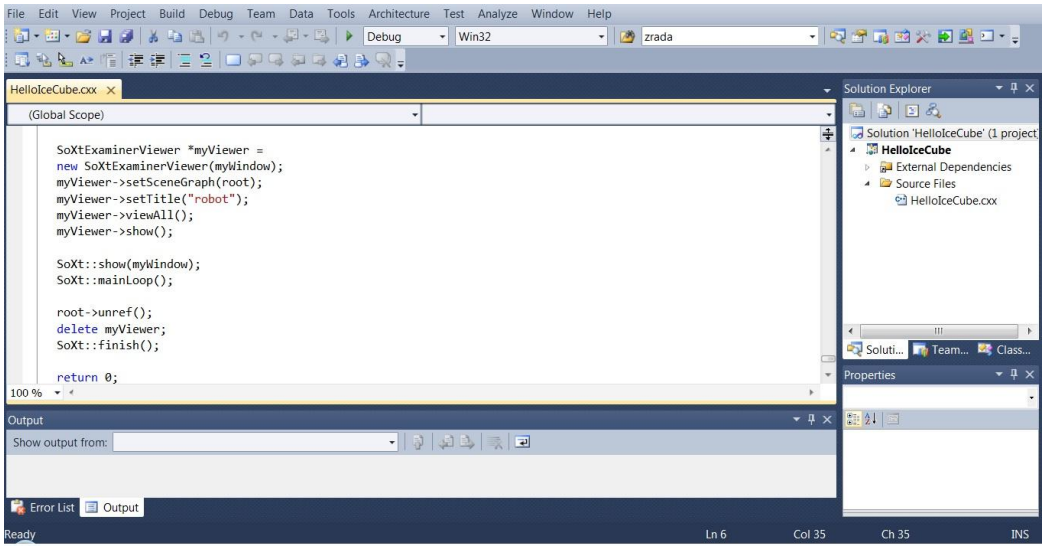
- *Blinker engines* - izravno mijenja sveobuhvatno stanje na grafu scene uzrokujući da se pojedini dijelovi scene periodično ponavljaju. Najočitija uporaba je izazvati predmet da se pojavi nekoliko puta i nestane iz vida, stoga ima naziv "žmigavac" (eng. „Blinker“). Ove vrste čvorova izvedene su iz čvorova *Switch* grupe, koji kontroliraju koji će točno dijelovi grafa scene biti aktivni. [5]

4. OPEN INVENTOR ZADATAK

4.1. Visual Studio 2010

Kako bi se koristila Open Inventor knjižnica algoritama na računalu mora postojati program u kojem je omogućeno korištenje C++ programskog jezika.

U ovom radu korišten je Visual Studio 2010 Ultimate Edition [Slika 21].



Slika 21 Korisnički prozor Visual Studio 2010

Visual Studio je razvojno okruženje (eng. „Integrated Development Environment“ tj. IDE) namijenjeno razvoju raznih vrsta aplikacija na Windows platformi.

Radi se o korisničkom sučelju (namijenjenim ugodnijem radu programera) povezanim s jezičnim procesorima koji podržavaju rad s raznim programskim jezicima. Najčešći su Microsoft Visual Basic, C++ i C#. Visual Studio podržava i dizajnerski pristup izradi programskog koda - onaj u kojemu kod ne treba pisati, nego se programiranje može obavljati raspoređivanjem grafičkih ikona koje simboliziraju dijelove koda pri čemu se upisuju samo neke osnovne značajke (eng. Properites) tih dijelova koda.

Mirosfot Visual Studio se može koristiti i kao alat za izradu web-aplikacija. Mogućnosti poput JavaScript IntelliSensa, Debugginga, CSS editora, CSS IntelliSensa, HTML editora ključne su prednosti Visual Studia pri razvoju web aplikacija.

Visual Studio je daleko više od jednostavnog editora namijenjenog pisanju izvornog koda. Konstantnim razvojem Visual Studio dobivao je nove mogućnosti u radu i razvoju aplikacija koje ga pozicioniraju kao sveobuhvatni alat za razvoj najraznovrsnijih aplikacija. [9]

4.2. Scena razvijena Open Inventor-om

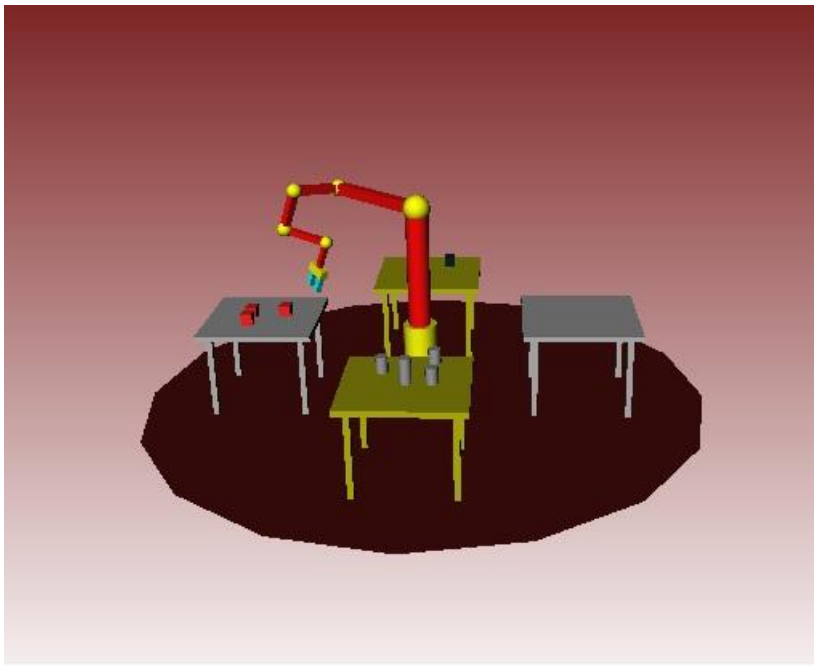
Zadatak ovog diplomskog rada bio je izraditi scenu koja sardži šestoosni manipulator s pripadajućom okolinom.

Scena izrađena pomoću Open inventor knjižnice algoritama i programa Visual Studio 2010 nalazi se na [Slika 22] i [Slika 23].

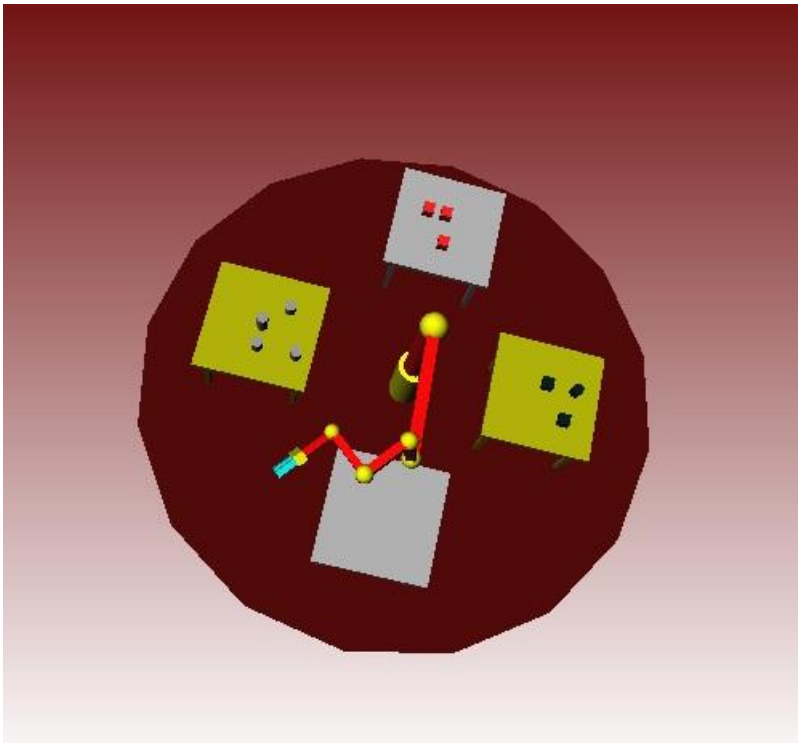
Scena je kreirana u sintaksi C++ programskog jezika. Ona se sastoji od sljedećih elemenata:

- Čvora kamere
- Čvora osvjetljenja
- Čvora pozadine
- Čvorova koji kreiraju jednostavne objekte scene (kvadar, valja, kugla...)
- Čvorova koji definiraju boje objekata scene
- Čvorova koji pozicioniraju objekte u sceni
- Čvorova koji rotiraju objekte u sceni
- Čvorova koji gibaju objekte u sceni
- Čvorovi koji prikazuju scenu na ekranu

Potrebno je poznavanje svih čvorova kako kako bih mogli stvoriti Open Inventor graf scene.
Potrebno je poznavanje hijerarhije Open Inventor grafa scena kako bi čvorovi kreirani u Open Inventor sintaksi koda na pravilan način utjecali jedan na druge.
Potrebno je podesiti polja čvorova Open Inventor scene kako bi željene vrijednosti bile primjenjene u sceni.



Slika 22 Scena kreirana u Open Inventoru



Slika 23 Scena kreirana u Open Inventoru, drugi pogled

5. IvTune EDITOR

IvTune je editor i preglednik koji koristi Open Inventor graf scene.

Može biti korišten za izradu, optimiziranje i otklanjanje pogrešaka (eng. „debugging“) unutar grafa scene. Editor dozvoljava uvid u strukturu grafa, mijenjanje varijabli i prikaz promjena koje se događaju na ekranu. Ovaj program korišten je za izradu scene s dva šestoosna manipulatora s pripadajućom opremom karakterističnom za poslove robotskog sklapanja.

Pristup editoru moguć je na dva načina:

- Korištenjejem programa, IvTuneViewer, koji omogućava pregled i uređivanje datoteka koje se nalaze u grafu scene. Program može biti korišten na Windows i Linux platformi.

- Iz pokrenute Open inventor aplikacije, u tom slučaju pregled i uređenje grafa scene se odvija unutar aplikacije.

Kada je IvTuneViewer pokrenut, otvara se prozor koji sadrži Open Inventor preglednik „Examiner Viewer“, alatnu traku izbornika i četiri prozora oko preglednika [Slika 24]. Preglednik „Examinar Viewer“ služi za interaktvno mijenjanje scene, dok drugi prozori služe za uređenje grafa scene. Ti prozori uključuju „Tree View“ koji prikazuje graf scene, „Node Overview“ koji prikazuje podatke o čvorovima (eng. „node“) i prozori „Field Editor“ i „Field Watch“ koji prate i mijenjaju vrijednosti unutar Open inventor polja (eng. „fields“).

Ako program IvTune pokrenemo iz aplikacije, otvara nam se poseban prozor. U njemu se, isto tako, nalazi prozor „Tree View“ s hijerarhijski prikazanim čvorovima koji tvore scenu, „Field Watch“ i „Field Editor“ s podacima o poljima koje koristi Open Inventor, ali aplikacija koristi svoj vlastiti preglednik i on ne mora biti „Examinar Viewer“.

Dva su načina da pokrenemo IvTune iz Open Inventor aplikacije ili programa:

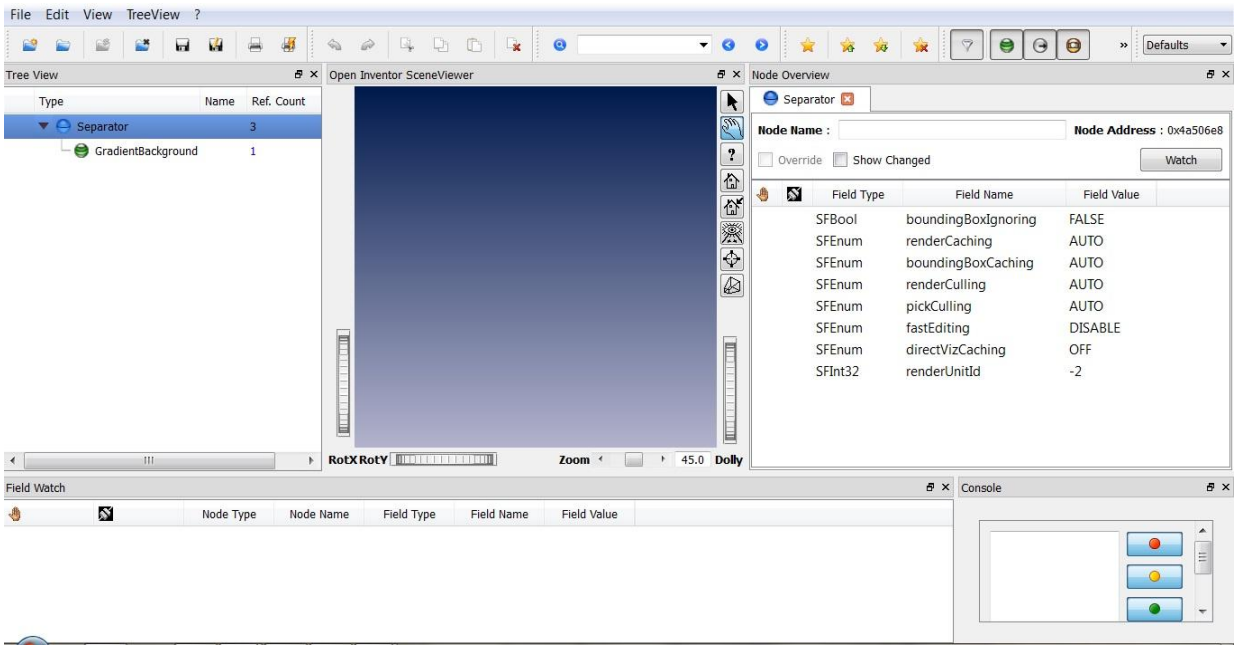
- U programskom kodu može se koristiti sintaksa: `SoIvTune::setInstance()`

- IvTune možemo pokrenuti i ručno, kursor treba postaviti u prozor preglednika i pritisnuti na tipkovnici prečicu SHIFT+F12

U nastavku rada bit će prikazani osnovni principi rada sa IvTuveViwer-om, način na koji se kreiraju čvorovi, njihova polja i prezentiran rad koji je napravljen u tom editor-u.

Nakon početnog pokretanja IvTune editora otvara se preglednik prikazan na [Slika 24]. U njemu se nalaze, kao što je već bilo riječi, „Examiner Viwer“ u sredini preglednika i 4 prozora oko preglednika:

- Tree View ([Slika 24],lijevo)
- Node Overview ([Slika 24],desno)
- Field Watch ([Slika 24], dolje lijevo)
- Field Editor ([Slika 24], dolje desno)



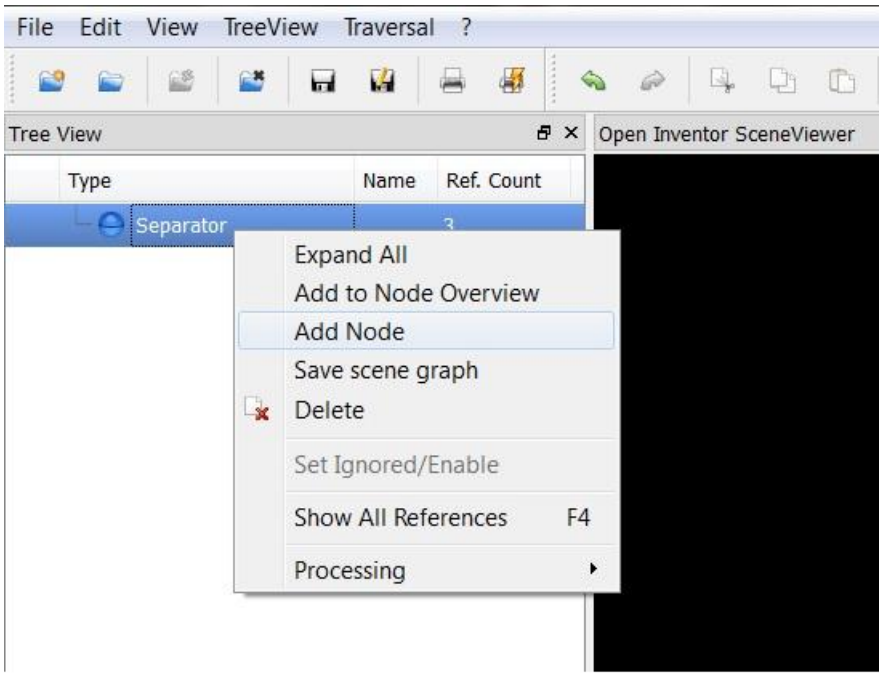
Slika 24 IvTune Viewer

Kod prvog otvaranja IvTune editora u prozoru „Tree View“, koji opisuje scenu, nalazi se grupacijski čvor *Separator* i unutar njega čvor za oblikovanje pozadine scene *GradientBackground*.

U nastvaku će biti opisano na koji način se koristi IvTune Viewer, koje su njegove mogućnosti i na kraju, bit će prikazana i opisana scena s robotskim manipulatorima i pripadajućom okolinom. Kao i kod razvoja scene s programom Visual Studio 2010, korištenje IvTune Viewer-a je vrlo slično. I dalje se koriste Open Inventor čvorovi i polja, ali na drugačiji način. U ovom slučaju postoji interaktivno sučelje IvTune Viewer-a i stvaranje scene je olakšano. [10]

5.1. IvTune čvor

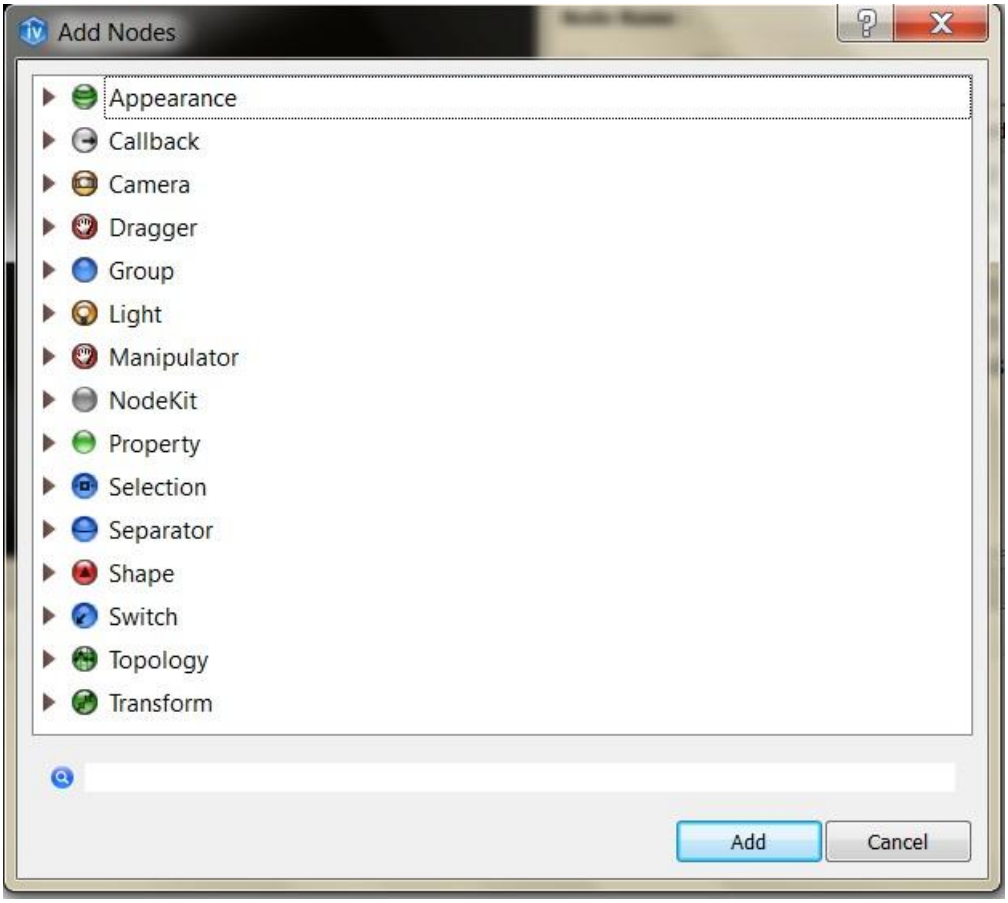
Čvorovi pretstavljaju klase i funkcije Open Inventor knjižnice algoritama. Kreiranje novog čvora izvršava se unutra prozora „Tree View“. „Tree View“ prozor sadržava sve čvorove,a oni svi zajedno čine graf scene. Kreiranje novog čvora vrši se na način da se mišem pomakne na čvor *Separator* [Slika 25], pritisne desni-klik i odabere „Add Node“. *Separator* čvorovi se koriste kako bi se dobila hijerarhijska struktura unutar grafa scene. Oni funkcioniraju kao „razdjelnici“ u smislu da se želi određene čvorove izolirati jedne od drugih. [11]



Slika 25 Dodavanje novog čvora

Nakon odabira naredbe „Add node“ pojavljuje se prozor prikazan na [Slika 25] . U njemu se nalaze grupe čvorova. Svaka grupa ima određenu namjenu.

Tako npr. grupa čvorova *Appearance* [Slika 26] sadrži čvorove koji stvaraju i mijenjaju određena svojstva scene poput boje pozadine, teksture itd., grupa *Shape* sadrži čvorove za oblikovanje različitih objekata (valjak, kugla, stožac…), dok grupa *Light* sadrži čvorove koji stvaraju i modificiraju osvjetljenje unutar scene. [11]



Slika 26 Grupe čvorova

Postoji 18 grupa čvorova u IvTune editoru. U daljnjem tekstu bit će pobliže opisane one grupe čvorova koje su korištene u ovom diplomskom radu.

5.2. Čvor *Separator*

Čvor Separator [Slika 27] nalazi se u istoimenoj grupi, a služi kako bi se stvorila hijerarhijska strukturu unutar grafa scene. On izolira pojedine čvorove kako bi neke akcije bile izolirane od ostatka grafa scene. U daljnjem tekstu dan je primjer korištenja čvora *Separator*.

Slika 27 Čvor *Separator*

Na [Slika 28] prikazan je primjer grafa scene koji sadrži glavni čvor *Separator* i u njemu dva čvora, *Lijevi_Separator* i *Desni_Separator*. Čvor *Lijevi_Separator* sadrži čvor *Cone*, to je čvor za oblikovanje stožca. Čvor *Desni_Separator* sadrži čvor *Cube* koji služi za oblikovanje kocke ili kvadra.

U svakom čvoru nalazi se čvor *Material* koji utječe na svojstva ostalih čvorova, u ovom slučaju čvorovi utječu na boju stožca u *Lijevom_Separator-u* i boju kugle u *Desnom_Separator-u*.

Na ovaj način dobivamo dva izolirana sustava, jedan u čvoru *Lijevi_Separator*, jedan u čvoru *Desni_Separator*.

Slika 28 Primjer grafa scene sa *Separator* čvorovima

Kada ta dva čvora ne bi postojala [Slika 29] čvor *Material* (boja_stožca) bi utjecao na sve čvorove ispod njega zbog toga što se nalazi u istom čvoru *Separator* kao i ostatak čvorova scene. On bi utjecao i na čvor *Stožac* i na čvor *Kocka*. Čvor *Material* (boja_kocke) bi utjecao samo na čvor *Kocka*, zbog toga što se samo taj čvor nalazi ispod njega.

Slika 29 Čvorovi unutar jednog čvora *Separator*

U oba slučaja scena bi bila ista. Čvor *Material* (boja_kocke) nalazi se neposredno iznad čvora *Kocka* pa čvor *Material* (boja_stožca), iako utječe na čvor *Kocka*, gubi smisao.

Kada bi bio izbačen čvor *Material* (boja_kocke) dobio bi se graf scene prikazan na [Slika 30]. U ovom slučaju čvor *Material* (boja_stožca) bi imao utjecaj na oba čvora ispod njega i boja koju bi se postavila odnosila bi se i na čvor koji kreira stožac i na čvor koji kreira kocku.

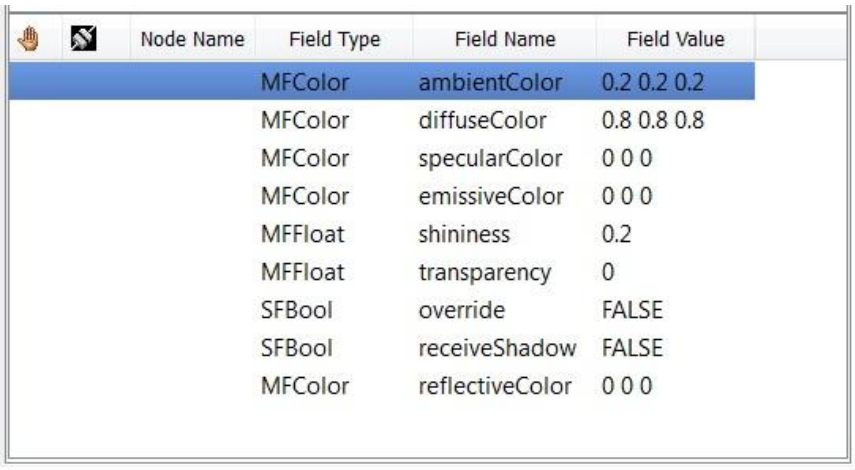
Slika 30 Graf scene bez čvora *Material* (boja_kocke)

Vidi se da kao i kod programiranja u C++ sintaksi i ovdje postoji „vertikalna hijerarhija čvorova“, što znači da čvorovi kreirani prvi unutar jednog čvora *Separator* imaju utjecaj na sve čvorove ispod sebe.

5.3. Čvor *Material*

Scena bi bila prilično jednolična ako bi svi objekti scene imali isti izgled. Najjednostavniji način na koji možemo promijeniti izgled objekata je pomoću čvora *Material.*

Polja klase *Material* prikazana su na [Slika 31] .Većina polja ove klase predstavljaju upravljanje bojom objekata na različite načine. Boja je predstavljena kao skup od tri broja npr. „0.0 0.4 0.8“. Brojevi iskazuju zasičenost crvenom, zelenom i plavom bojom, a boja prikazana na ekranu je miješavina sve tri boje. Iznos 0.0 znači da se jedna od boja uopće ne koristi, dok iznos 1.0 znači da se koristi 100% zasičenosti dotične boje. Stvaranje boje u sceni može biti znanost sama po sebi. Najjednostavniji način je koristiti samo polje *ambientColor* [Slika 31] kako bi se postavila boja određenog objekta scene. Na boju scene utječe i osvjetljenje o čemu će biti riječi kasnije. [11]



Slika 31 Polja čvora *Material*

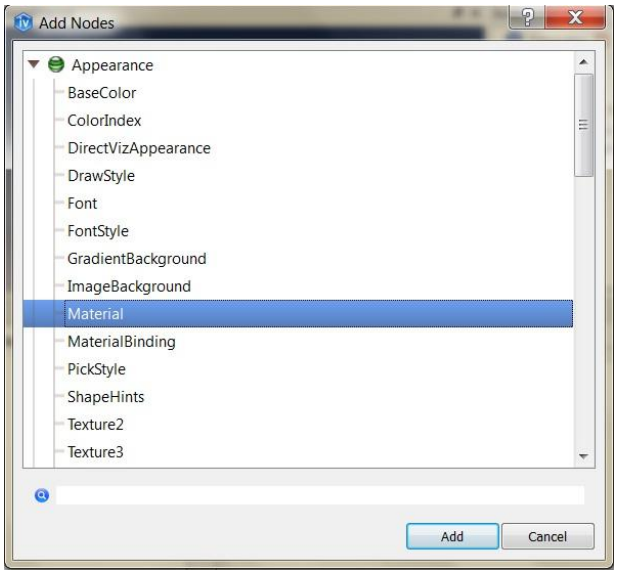
Postoji 9 polja unutar čvora *Material* [Slika 31]. Svako polje ima svoj tip (eng. „Field type“). Prefiks svakog tipa polja sadrži SF ili MF. SF dolazi od eng. „Single Value Field“ što znači da polje tog tipa ima samo jednu vrijednost npr. „TRUE“ ili „FALSE“. MF (eng. „Multiple Value Field“) je polja koje sadržava više vrijednost. [11]

Čvor Material ima tri tipa polja, to su [11]:

- *MFColor* - polje koje sadržava tri vrijednosti crvene, zelene i plave boje
- *MFFloat* - polje koje sadržava jednu vrijednost u obliku decimalnog broja
- *SFBool* – polje tipa „TRUE“ ili „FALSE“

Polja čvora utječu na mnoga svojstva objekata scene poput boje, prozirnosti, svjetlucavosti...

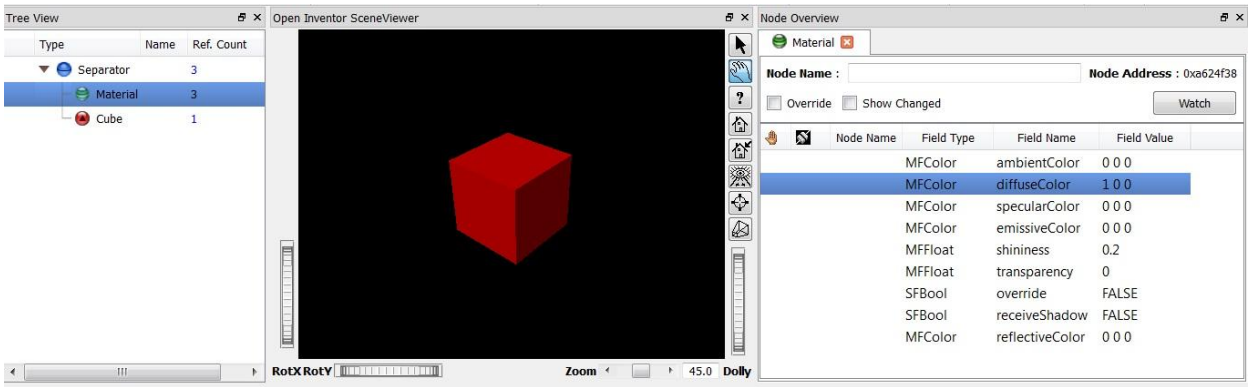
U IvTune editoru čvor *Material* nalazi se u grupi *Appearance* [Slika 32]



Slika 32 Grupa *Appearance*

Treba napomenuti da se čvor *Material* uvijek stavlja ispred čvorova za oblikovanje objekata. To je zato što on djeluje samo na čvorove koji su hijerarhijski ispod njega.

Primjer jednostavne scene koja sadrži jedan čvor *Separator* i unutar njega čvor *Material* koji djeluje na čvor *Cube* nalazi se na [Slika 33]. Čvor *Cube* sadrži klase za oblikovanje objekata u obliku kocke ili kvadra i on će biti opisan kasnije. [11]

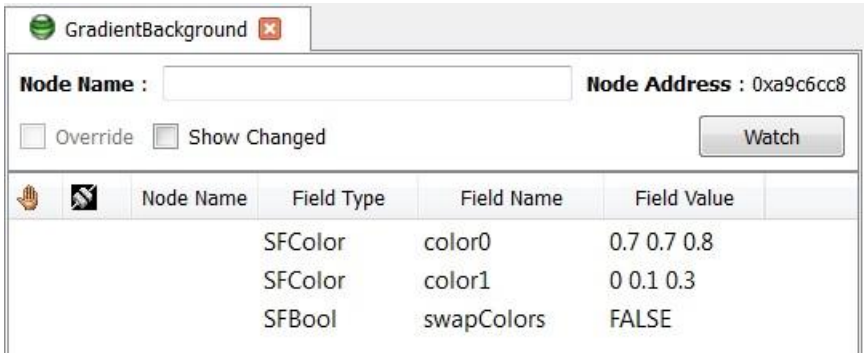


Slika 33 Primjer korištenja čvora *Material*

Kako bi kocka bili crvena treba promijenili polje *diffuseColor* [Slika 33] u vrijednost 1,0,0.

5.4. Čvor *GradinetBackground*

Još jedan čvor korišten u izradi ovog diplomskog rada je čvor *GradinetBackground*. On služi za oblikovanje pozadine scene. Pozadina ima vertikalne gradijente dvije boje po izboru korisnika. Prva boja ima svoje gradijente od vrha prozora od sredine, a druga od sredine do donjeg ruba prozora scene. Polja čvora *Gradientbackground* nalaze se na [Slika 34] . [11]

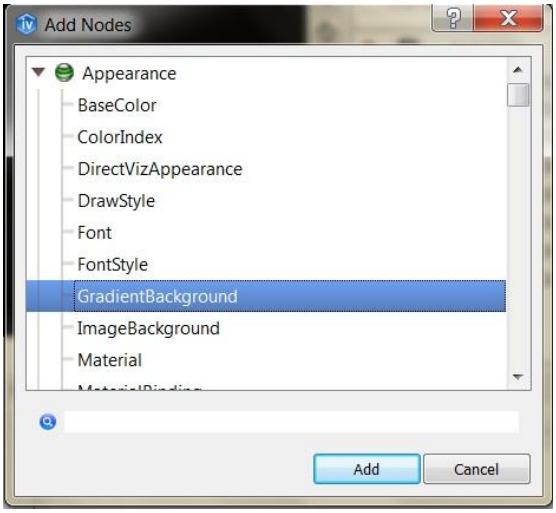


Slika 34 Polja čvora *GradinetBackground*

Postoje tri polja [11]:

- *color0* – definira prvu boju pozadine, ona se pojavljuje u doljnjem dijelu scene
- *color1* – definira drugu boju pozadine, ona se pojavljuje u gornjem dijelu scene
- *swapColor* – ako ga postavimo u „TRUE“ mijenja mjesta prve i druge boje pozadine

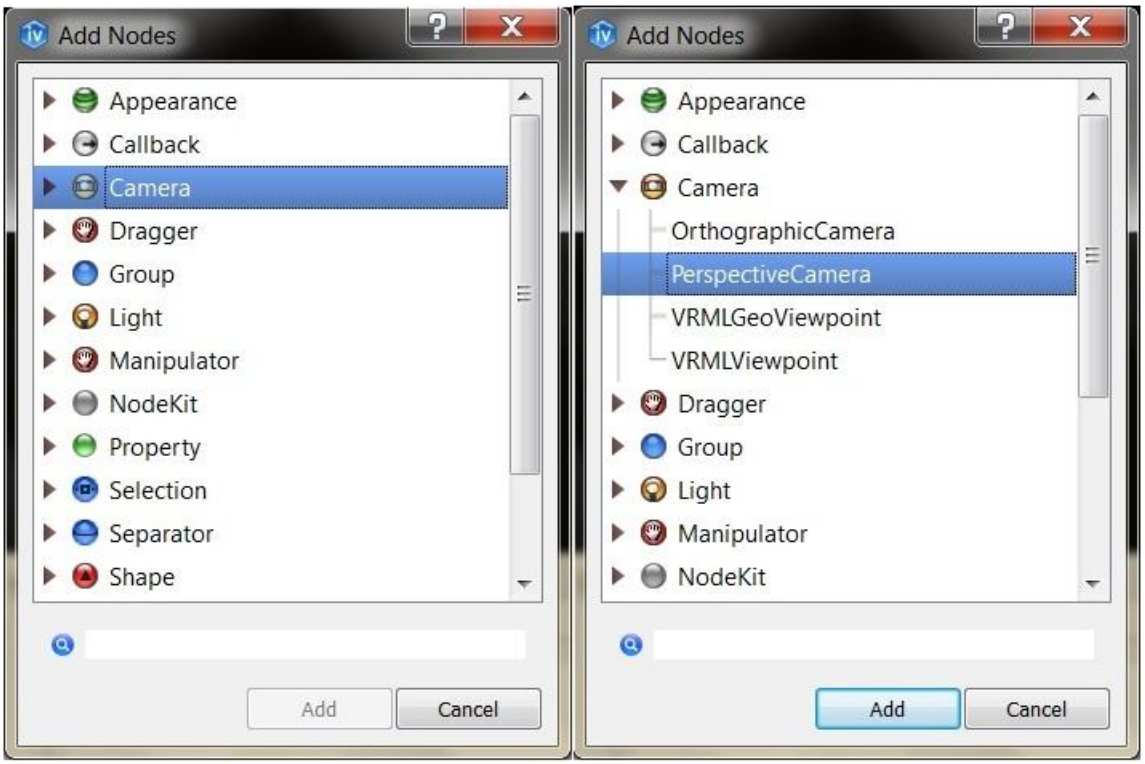
U IvTune editoru čvor *Material* nalazi se u grupi *Appearance* [Slika 35].



Slika 35 Čvor *GradientBackground* u grupi *Appearance*

5.5. Grupa *Camera*

Grupa *Camera* [Slika 36] sadrži čvorove koji prikazuju scenu u prozoru „Examinar Viewer“. Kako bi scena bila dobor vidljiva potrebno je dobro podesiti polja koja se nalaze unutar čvorova ove grupe. Ako se čvorovi grupe *Camera* ne nalaze u grafu scene IvTune editor postvalja unaprijed zadani pogled na scenu. [11]



Slika 36 Grupa *Camera*

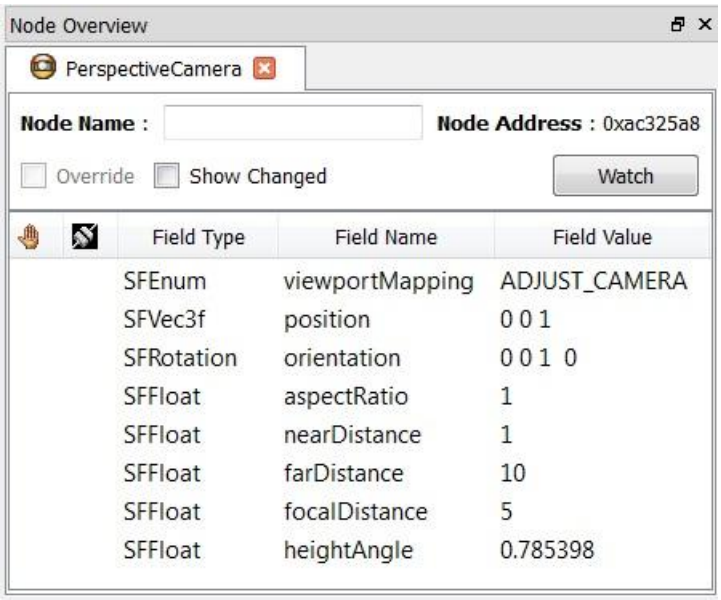
Najčešće korišteni čvorovi ove grupe su [11]:

- čvor *OrthographicCamera*
- čvor *PerspectiveCamera*

Razlika je u tome što čvor *PerspectiveCamera* daje pogled na scenu u kojem gledatelj ima dojam dubine scene. Čvor *OrthographicCamera* ne daje tu mogućnost.

5.5.1. Čvor *PerspectiveCamera*

Ovaj čvor koristi se u izradi pogleda na scenu. Njegova polja su prikazana u prozoru „Node Overview“ na [Slika 37]. [11]



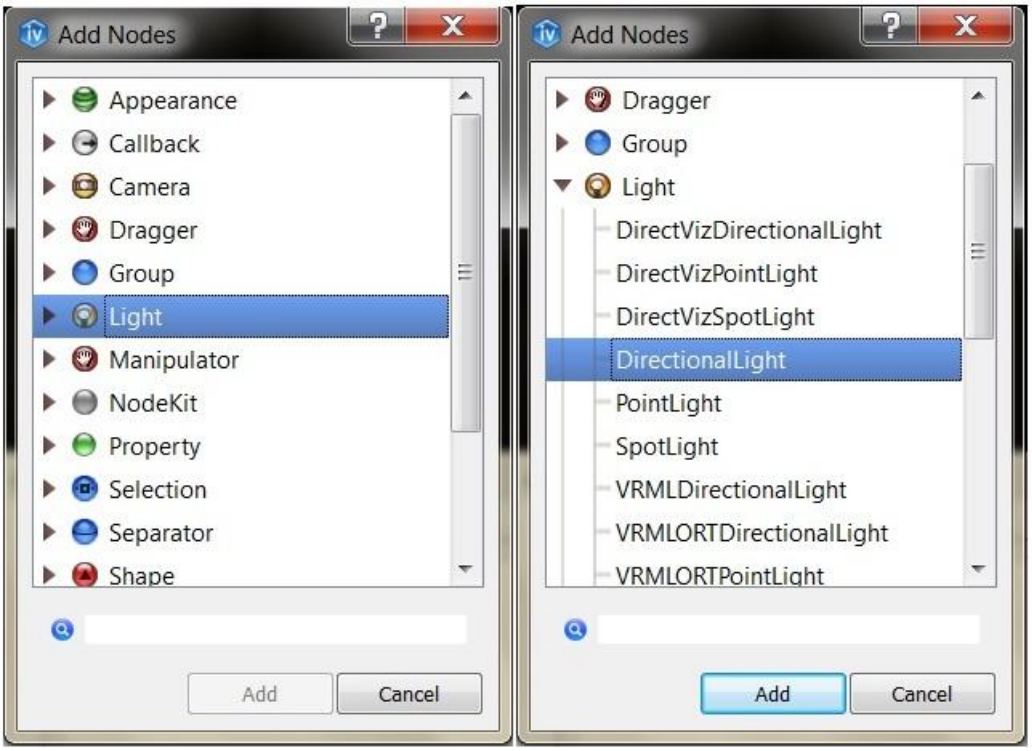
Slika 37 Polja čvora *PerspectiveCamera*

Polja čvora *PerspectiveCamera* su [11]:

- *viewportMapping* – daje različite poglede na scenu
- *position* – pozicionira pogled na scenu u odnosu na osi X,Y i Z
- *orientation* – rotira pogled na scenu u odnosu na os X,Y i Z
- *aspectRatio* – određuje omjer širine i visine pogleda kamere
- *nearDistance* – udaljenost najbližeg objekta od kamere
- *farDistance* – udaljenost kamere od scene
- *focalDistance* – određuje fokus kamere scene
- *heightAngle* – određuje kut (u radijanima) pod kojim stoji kamera koja gleda na scenu

5.6. Grupa *Light*

Grupa *Light* [Slika 38] služi za osvjetljenje scene. Čvorovi ove grupe imaju polja koja omogućuje uključivanje i isključivanje osvjetljenja, promjenu inteziteta i kuta osvjetljenja itd. [11]



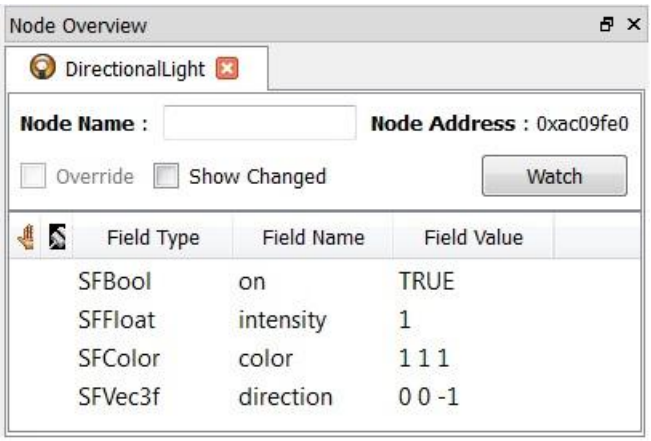
Slika 38 Grupa *Light*

Najčešće korišteni čvorovi ove grupe su [11]:

- *DirectionalLight* – omogućava osvjetljenje cijele scene, izvor osvjetljenja ima svoje koordinate i smjer, nešto poput reflektra
- *PointLight* – osvjetljenje dolazi iz jednog izvora u svim smjerovima, nešto poput svjetla svijeće
- *SpotLight* – objedinjuje svojstva prethodna dva čvora, osvjetljenje dolazi iz jednog izvora i širi se u obliku stožca

5.6.1. Čvor *DirectionalLight*

Ovaj čvor koristi se u izradi osvjetljenja scene. Njegova polja su prikazana u prozoru „Node Overview“ na slici [Slika 39].



Slika 39 Polja čvora *DiretionalLight*

Polja čvora *DiretionalLight* su [11]:

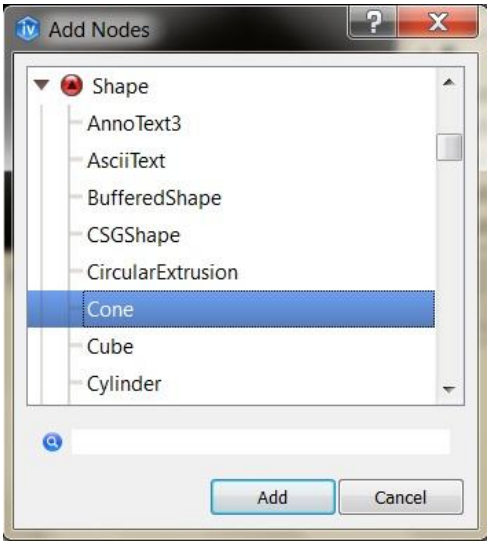
- *on* – polje tipa SFBool, sadrži vrijednost „TRUE“ ili „FALSE“, ukoliko je postavljeno na „FALSE“ čvor *DirectionalLight* nije aktivan
- *intensity* – polje tipa SFFloat, sadrži vrijednost jačine osvjetljenja u obliku decimalnog broja, poprima vrijednosti od 0 do 1
- *color* – polje tipa SFColor, sadrži tri vrijednosti koje se odnose na intenzitet crvene, zelene i plave boje
- *direction* – postavlja osvjetljenje s obzirom na os X,Y i Z, osvjetljenje je usmjereno prema središtu scene

5.7. Čvorovi grupe *Shape*

Čvorovi unutar grupe *Shape* sadrži objekte koji mogu biti prikazani u sceni IvTune editora [11].

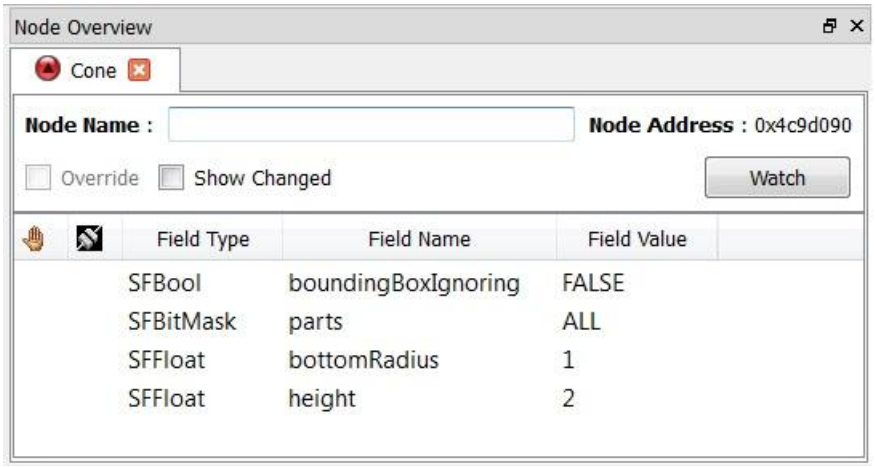
5.7.1. Čvor Cone

Jedan od čvorova u gupi *Shape* je čvor *Cone*, slika [Slika 40].



Slika 40 Grupa *Shape*

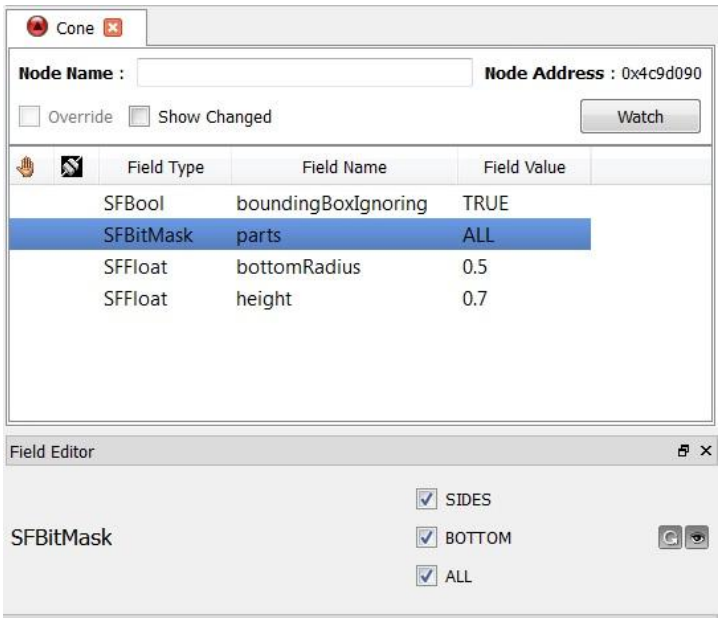
Čvor *Cone* služi za kreiranje jednostavnih objekata u obliku stožca. On sadrži 4 polja. Polja su prikazana u prozoru „Node Overview“ [Slika 41].



Slika 41 Polja čvora Cone

Polja čvora *Cone* su [11]:

- *boundigBoxIgnoring* – polje tipa SFBool sadrži vrijednosti „TRUE“ ili „FALSE“, prilikom postavljanje čvora *Cone* ovo polje ima vrijednost „FALSE“, što znači da editor IvTune ne postavlja koordinatni sustav u središte najmanjeg virtualne kvadra koja opisuje stožca, ako ovo polje postavimo u vrijednost „TRUE“ generira se koordinatni sustav koji možemo koristiti za određene namjene
- *parts* – pritiskom miša na polje parts otvara se prozor „Field Editor“ [Slika 42]



Slika 42 Polja *parts* i prozor „Field Editor“

Ovo polje je tipa SFBitMask i sadrži tri opcije za izbor prikaza vidljivih dijelova stožca na ekranu. Opcije za prikaz su:

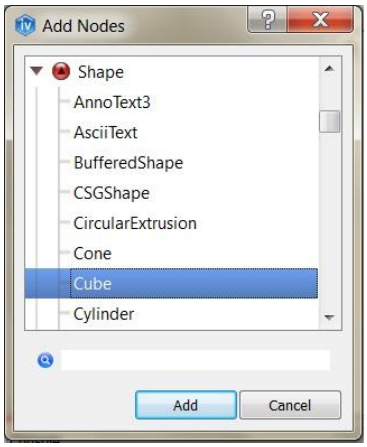
- SIDES – prikazuje samo plašt stožca, ali ne i bazu
- BOTTOM – prikazuje samo bazu stožca
- ALL – prikazuje i bazu i plašt stožca

Ovo polje ima, prilikom pokretanja, označene sve tri opcije prikaza tako da je stožac vidljiv u cijelosti.

- *bottomRadius* – pomoću ovog polja je moguće mijenjati radijus baze stožca
- *height* – ovim polje utječe na promjenu visine stožca

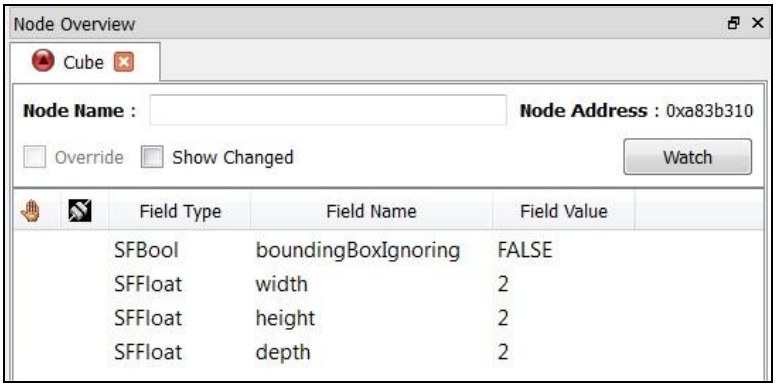
5.7.2. Čvor *Cube*

Čvor *Cube* još jedan je čvor u grupi *Shape* [Slika 44].



Slika 43 Grupa *Shape* i čvor *Cube*

Čvor *Cube* služi za kreiranje objekata u obliku kocke ili kvadra. On sadrži 4 polja. Polja su prikazana u prozoru „Node Overview“ [Slika 44].



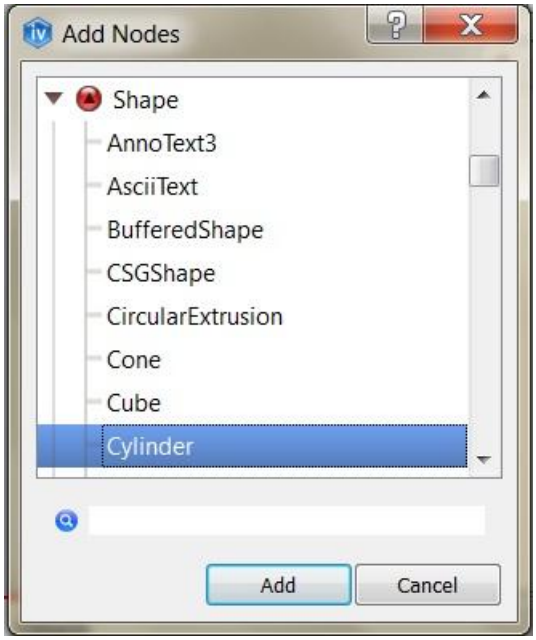
Slika 44 Polja čvora *Cube*

Polja čvora *Cube* su [11]:

- *boundingBoxIgnoring* – tipa SFBool sa vrijednostima „TRUE“ i „FALSE“ kao i kod čvora *Cone* postavlja koordinatni sustav u središte kocke ili kvadra, na početku je vrijednost ovog polja postavljena „FALSE“
- *width* – širina kocke ili kvadra
- *height* – visina kocke ili kvadra
- depth – dubina kocke ili kvadra

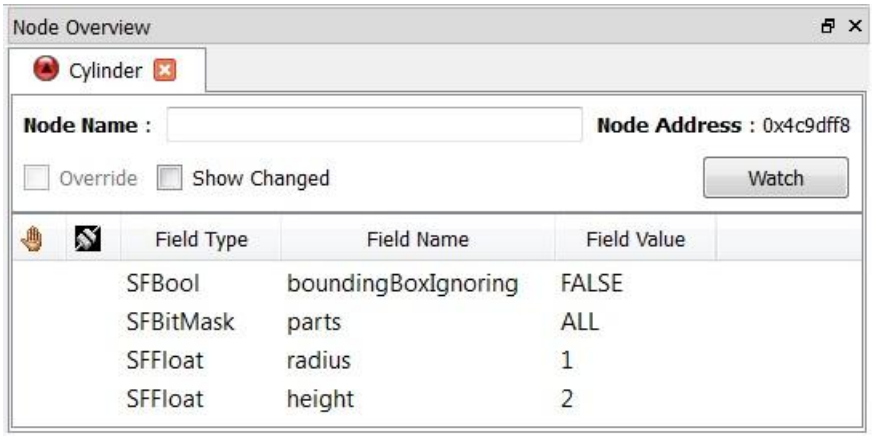
5.7.3. Čvor *Cylinder*

Još jedan čvor u grupi *Shape* korišten u ovom diplomskom radu [Slika 45].



Slika 45 Grupa *Shape* i čvor *Cylinder*

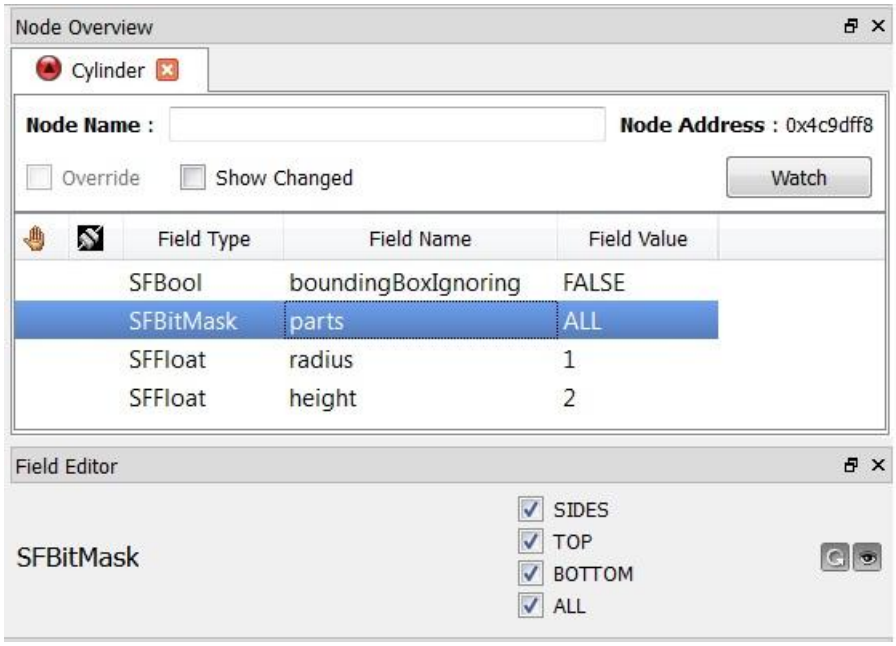
Čvor *Cylinder* služi za kreiranje objekata u obliku valjka. On sadrži 4 polja. Polja su prikazana u prozoru „Node Overview“ [Slika 46].



Slika 46 Polja čvora Cylinder

Polja čvora *Cylinder* su [11]:

- *boundingBoxIgnoring* – kao i kod predhodna dva čvora, ovo polje je zaduženo za postavljanje koordinatnog sustava u središte valjka, ovo polje u početku postavljeno na vrijednost „FALSE“
- *parts* – pritiskom miša na ovo polje otvara nam se prozor „Field Editor“ [Slika 47].



Slika 47 Polje *parts* i „Filed Editor“

Pozor „Field Editor“ sadrži opcije:

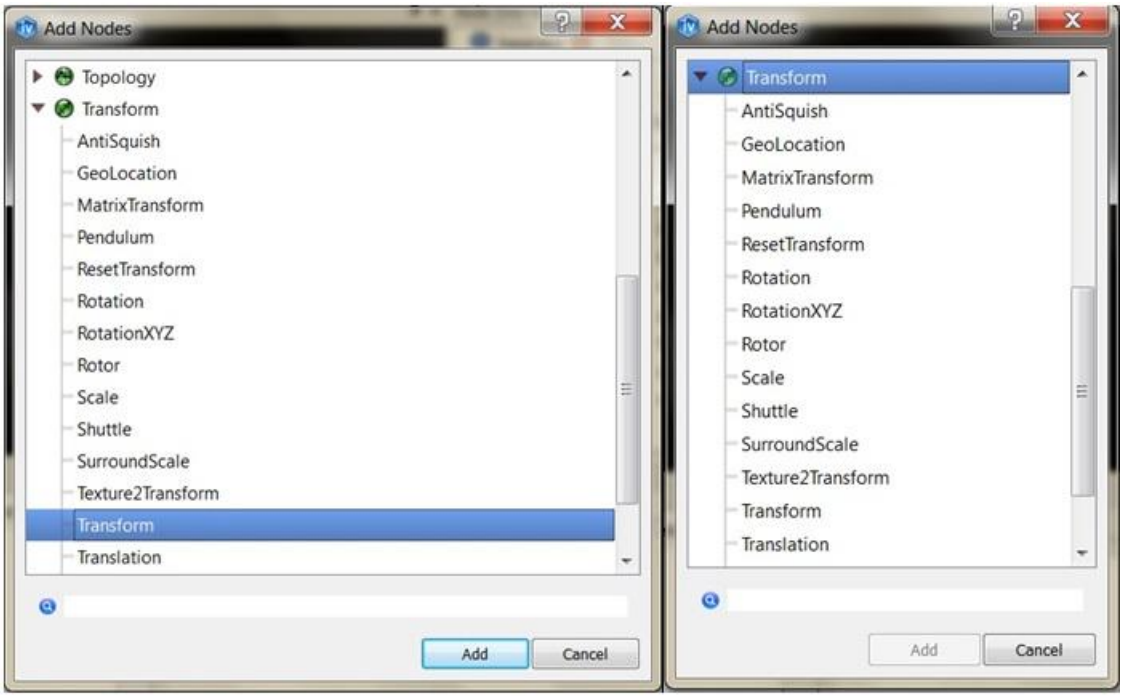
- SIDES – prikazuje samo plaš valjka
- TOP – prikazuje samo gornju bazu valjka
- BOTTOM – prikazuje samo donju bazu valjka
- ALL – prikazuje cijeli valjak

Kod kreiranja čvora *Cylinder* polje *parts* postavljeno je u vrijednost ALL i cijeli valjak je vidljiv u sceni.

- *radius* – polje koje sadrži radijus baze valjka
- *height* – visina valjka u sceni

5.8. Grupa *Transform*

Grupa *Transform* [Slika 48] sadrži čvorove koji mijenjaju geometrijska svojstva objekata u sceni. Najčešće su to veličina i pozicija. [11]



Slika 48 Grupa *Transform*

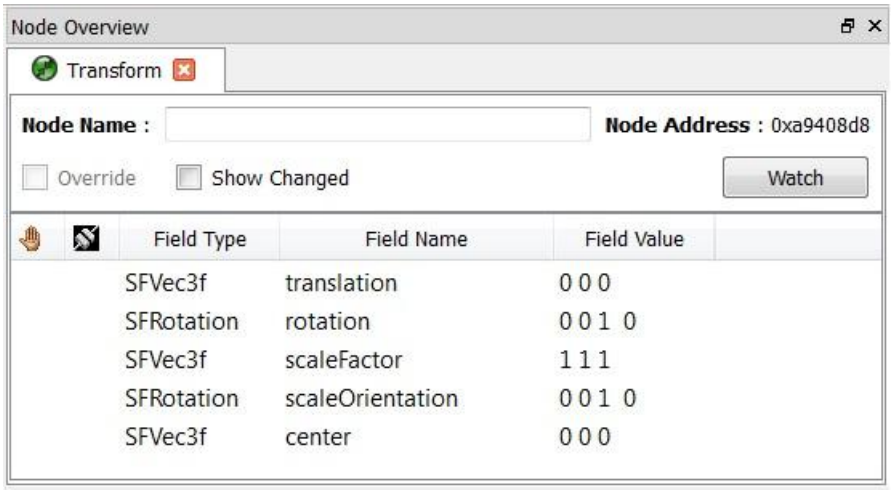
U nastvaku će biti opisani neki od najčešće korištenih čvorova iz ove grupe.

To su:

- Čvor *Transform*
- Čvor *Translation*
- Čvor *Rotation*
- Čvor *RotatioXYZ*

5.8.1. Čvor *Transform*

Čvor *Transform* nalazi se u istoimenoj grupi čvorova, grupi *Transform*. Ovo je najčešće korišten čvor za transformaciju objekata u sceni. Polja čvora *Transform* prikazana su na [Slika 49].



Slika 49 Polja čvora *Transform*

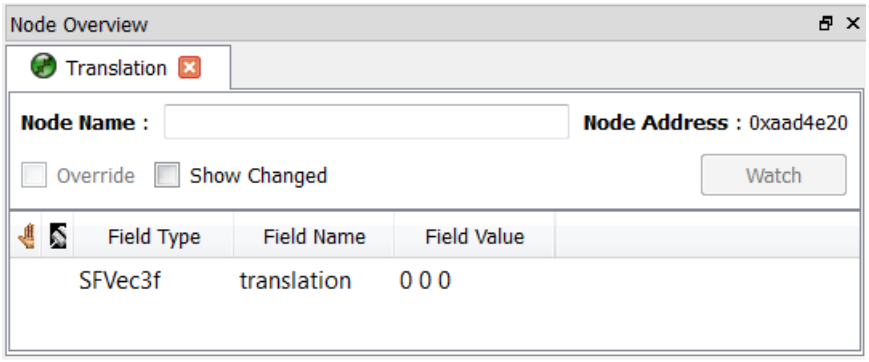
Polja čvora *Transform* su [11]:

- *translation* – polje tipa SFVec3f što označava trodimenzionalni vektor, polje za unos vrijednosti (eng. „Field Value“) ima 3 broja (npr. 1 0 2) što označava pomak u smjeru X,Y i Z osi.
- *rotation* – polje tipa SFRotation, polje za unos vrijedosti ima oblik npr. „0 0 1 3.12“, prva tri broja mogu poprimiti vrijednosti 0 ili 1, 1 označava os oko koje se vrši rotacije, a zadnji broj je iznos rotacije u radijanima.
- *scaleFactor* – polje tipa SFVec3f, polje za unos vrijednosti ima 3 broja koja označavaju veličinu objekta u smjeru osi X,Y i Z. Početni oblik je „1 1 1“, a promjenom vrijednosti nekog broja objekt u sceni se razmjerno povećava ili smanjuje
- *scaleOrientation* – polje tipa SFRotation, sadrži orijentaciju objekta koja će biti podvrgnuta postupku skaliranja
- *center* – translatiraju koordinatni sustav koji kasnije koriste polja scaleFactor i rotapku

5.8.2. Čvor Translation

Čvor *Translation* koristi se kada želimo translatirati objekt u našoj sceni, iako to možemo i sa čvorom *Transform*, čvor *Translation* zauzima manje mjesta u memoriji programa.

Polje čvora *Translation* nalaze se u prozoru „Node Overview“ (slika …).

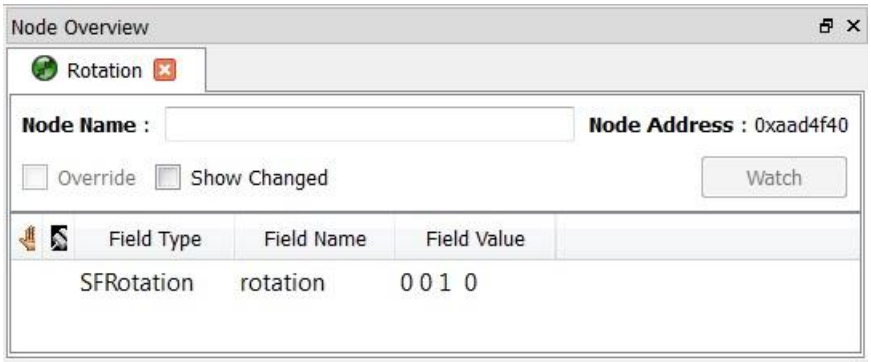


Slika 50 Polja čvora Translation

Polje *translation* je polje tipa SFVec3f i sadrži vrijednost od 3 broja, svaki broj predstavlja traslantaciju objekta u scenu u smjeru osi X,Y i Z. [11]

5.8.3. Čvor Rotation

Služi za rotaciju objekta u sceni oko osi X,Y i Z. Kao i kod prethodnog čvora i ovaj čvor možemo zamijeniti sa čvorom *Transform*, ali čvor *Rotation* zauzima manje mjesta u memoriji programa. Polje čvora Rotation je prikazano na [Slika 51].

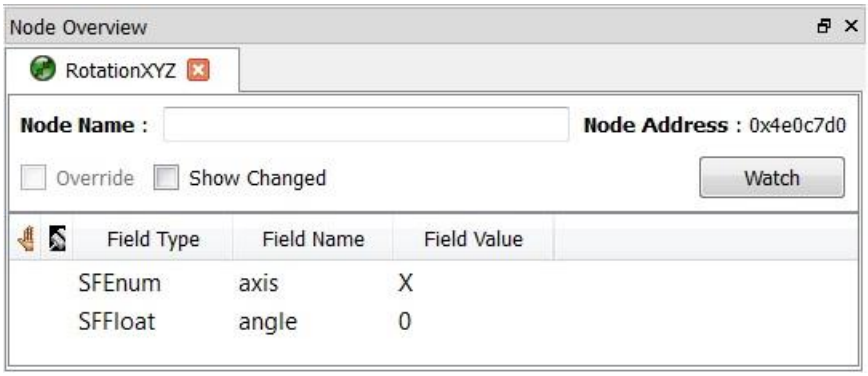


Slika 51 Polje čvora *Rotation*

Polje rotation ima 4 vrijednosti, prve tri vrijednosti mogu poprimiti vrijednost 1 ili 0 i označavaju os vrtnje, a zadnja vrijednost označava iznos rotacije u radijanima. [11]

5.8.4. Čvor RotationXYZ

Još jedan čvor za rotiranje objekata u sceni.Polja čvora *RotationXYZ* nalaze se na slici [Slika 52].



Slika 52 Polja čvora *RotationXYZ*

Polja čvora *RotationXYZ* su [11]:

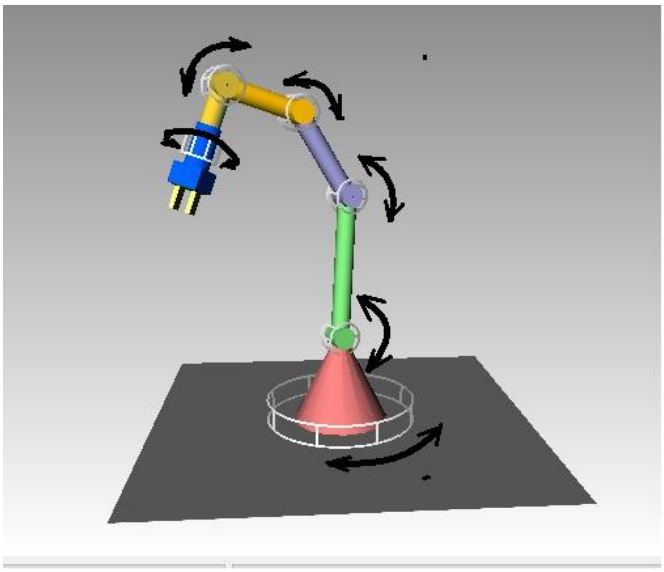
- *axis* – polje tipa SFEnum, sadrži izbornik u kojem možemo odabrati os X,Y ili Z oko koje će biti izvršena rotacije
- *angle* – polje tipa SFFloat, sadrži vrijednost ratacije u radijanima

5.9. Grupa *Dragger*

Još jedna grupa čvorova je grupa *Dragger*. To je grupa čvorova koja se koristi u grafu scene za specijalnu namjenu, komunikaciju sa kompjuterskim mišem.

Ona služi za kreiranje vidljivih objekata u sceni koji vruše određene operacije. Najčešće su to translacije, rotacije i skaliranja objekata scene. Operacije se izvršavaju na način da korisnik pritisne klikom miša na geometriju čvora grupe *Dragger* i pomicanjem uzroku neku promjenu objekata u grafu scene.

Primjer čvora koji je korišten u ovom diplomskom radu je čvor *RotateCylindricalDragger*. Taj čvor utječe na objekte grafa scene na način da ih rotira oko zadane točke. Na [Slika 53] prikazan je robotski manipulator sa 6 stupnjeva slobode, on također ima 6 geometrija čvora *RotateCylindricalDragger* koje omogućuju rotacije oko zglobova manipulatora. Članci manipulatora mogu biti rotirani u oba smjera kao što pokazuju strelice. [11]



Slika 53 Primjer korištenja čvora *RotateCylindricalDragger*

Najčešće korišteni čvorovi ove grupe su [11]:

- *CenterballDragger*
- *Dragger*
- *RotateCylindricalDragger*
- *TransformDragger*
- *Scale2Dragger*
- *Translate2Dragger*

6. IvTune ZADATAK

Nakon što su objašnjene sve grupe i svi pripadajući čvorovi Open Inventor grafa scene, moguće je kreirati funkcionalnu 3D scenu.

Na [Slika 54] i [Slika 55] prikazana je scena koja sadrži 2 šestoosna robotska manipulatora i pripadajuću okolinu.

Scena je kreirana pomoću Open Inventor IvTune editora.

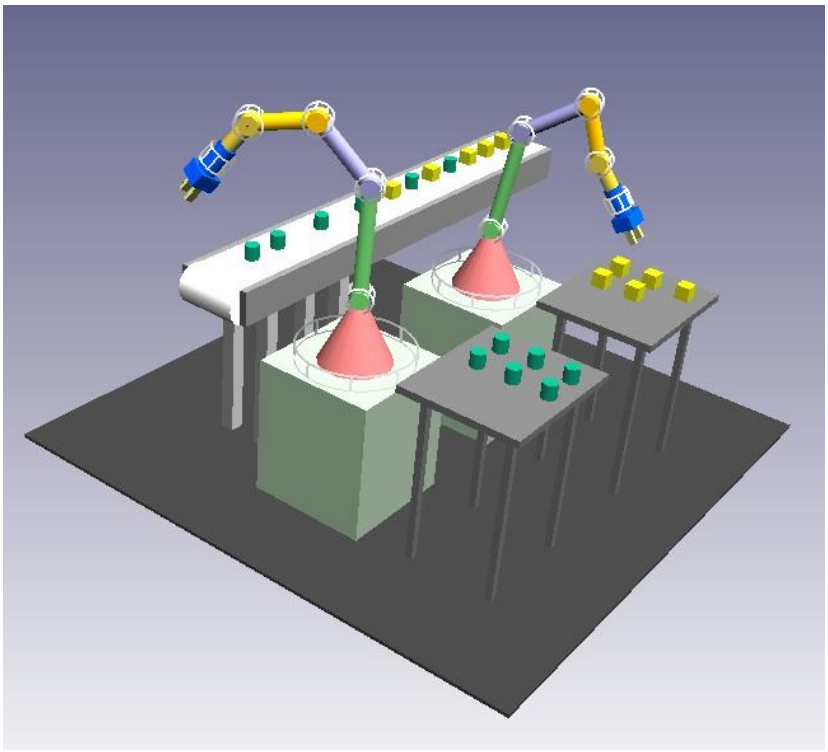
Scena se sastoji od preko 200 čvorova. Potrebno je poznavanje svih čvorova koji mogu biti kreirani s IvTune ediotorom kako bi scena bila uspješno razvijena.

Vrlo je važno znati hijerarhiju čvorova pripadajućeg grafa scene.

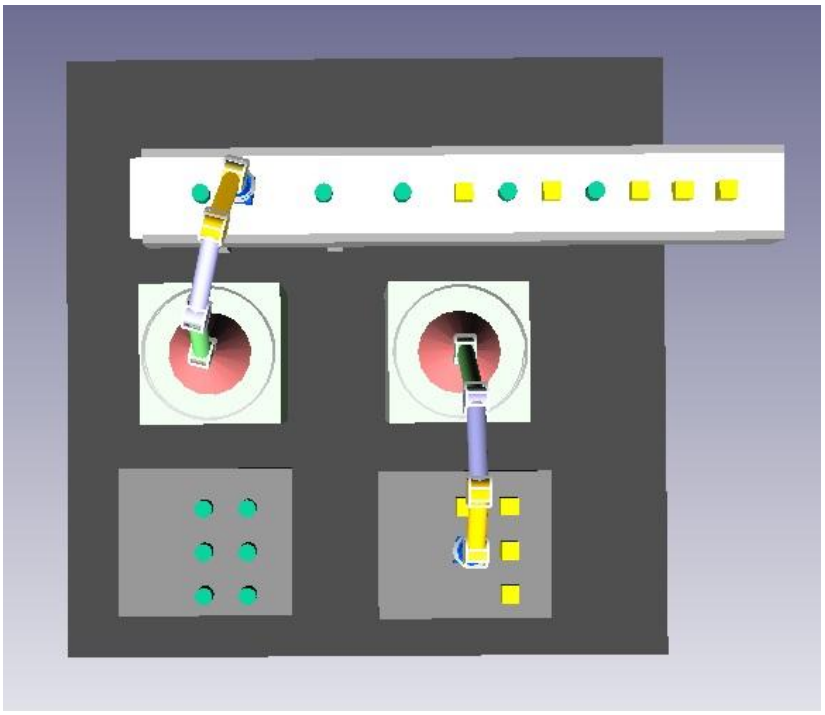
Scena na [Slika 54] i [Slika 55] sadrži:

- Čvor pozadine
- Čvor kamere
- Čvorove osvjetljenja
- Čvorove za pozicioniranje objekata
- Čvorove za rotiranje objekata
- Čvorove za manipulaciju objektima
- Čvorove za definiranje boje obekata

Potrebno je podesiti sva polja unutar prozora „Node Overview“ kako bi scena bila funkcionalna.



Slika 54 Prikaz 2 manipulatora i pripadajuče okoline



Slika 55 Prikaz 2 manipulatora i pripadajuče okoline (odozgo)

7. ZAKLJUČAK

Open Inventor je 3D grafička knjižnica s kojom je moguće kreirati nevjerojatnu raznovrsnost 3D aplikacija, uključujući aplikacije za geološke znanosti, tehnički dizajn i simulacijske aplikacije, medicinske aplikacije i aplikacije za virtualnu stvarnost. Zbog svoje jednostavnosti, Open Inventor-om se može uštedjeti puno vremena u razvoju 2D i 3D aplikacija jer se radi na višoj razini nego s prvobitnom OpenGL knjižnicom algoritama. Mnoge operacije koje programeri čine, kao što je pomicanje kamere i odabir objekata, su već uključene u Open Inventor. Također postoje mnoga proširenja koja omogućuju programiranje na naprednoj razini. Cilj ovog diplomskog rada bio je istražiti mogućnosti Open Inventor knjižnice algoritama i razviti šestoosni manipulator s pripadajućom okolinom za poslove robotskog sklapanja. Mnogo toga ostalo je još neprimjenjeno. Korak naprijed bio bi povezivanje virtualno stvorenog robotskog manipulatora s postojećim, „stvarnim“, robotom u laboratoriju. Na taj način ono što se događa na računalu bilo bi povezano sa sa stvarnošću. Open Inventor vrlo je moćan alat koji ima preko 1200 različitih klasa. Široko je primjenjiv i stalno se razvija, stvarajući nove klase i funkcije za specijalne namjene. Kako je značaj i korištenje računalne grafike i kreiranja virtualne stvarnosti sve veći u svijetu tako i ova knjižnica algoritama za kreiranje 3D scene ima sve više korisnika.

LITERATURA

[1] http://en.wikipedia.org/wiki/Open_Inventor

[2] http://www.vsg3d.com/open-inventor/sdk

[3] http://oivdoc86.vsg3d.com/node/18793

[4] http://www.vsg3d.com/open-inventor/extensions

[5] http://www-evasion.imag.fr/~Francois.Faure/doc/inventorMentor/sgi_html/

[6] http://doc.coin3d.org/Coin-1/index.html

[7] http://oivdoc92.vsg3d.com/node/15927

[8] http://www.vsg3d.com/support/oiv_doc/TechDoc/PG-GettingStarted.pdf

[9] http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[10] http://www.vsg3d.com/support/oiv_doc/ReferenceManual/html/toolsivtuneviewer.html

[11] http://web.mit.edu/ivlib/www/iv.html

PRILOZI

I. CD-R disk sa:

- diplomski rad u PDF formatu
- VC++ projekt diplomskog rada izrađen sa Visual Studiom 2010
- .iv file sa diplomskim radom izrađenim s IvTune editorom
- datoteka sa slikama korištenim u diplomskom radu